'Managed Challenge' Alleviates Disengagement in Co-evolutionary System Identification

Josh C. Bongard
Computational Synthesis Laboratory
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY 14850 USA
josh.bongard@cornell.edu

Hod Lipson
Computational Synthesis Laboratory
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY 14850 USA
hod.lipson@cornell.edu

ABSTRACT

In previous papers we have described a co-evolutionary algorithm (EEA), the estimation-exploration algorithm, that infers the hidden inner structure of systems using minimal testing. In this paper we introduce the concept of 'managed challenge' to alleviate the problem of disengagement in this and other co-evol-utionary algorithms. A known problem in co-evolutionary dynamics occurs when one population systematically outperforms the other, resulting in a loss of selection pressure for both populations. In system identification (which deals with determining the inner structure of a system using only input/output data), multiple trials (a test that causes the system to produce some output) on the system to be identified must be performed. When such trials are costly, this disengagement results in wasted data that is not utilized by the evolutionary process. Here we propose that data from futile interactions should be stored during disengagement and automatically re-introduced later, when the population re-engages: we refer to this as the *test bank*. We demonstrate that the advantage of the test bank is two-fold: it allows for the discovery of more accurate models, and it reduces the amount of required training data for both parametric identification – parameterizing inner structure – and symbolic identification – approximating inner structure using symbolic equations – of nonlinear systems.

Categories and Subject Descriptors

J.2 [**Computer Applications**]: Physical Sciences and Engineering

General Terms

Algorithms

Keywords

Co-evolution, System Identification

1. INTRODUCTION

Disengagement is one of several pathologies known to exist in artificial co-evolutionary systems [8] [22]. This occurs when either the individuals from one population are too easy, in which case individuals from a second population can all dominate them; or too difficult, in which case all the individuals from the second population are dominated by them. In either case, the fitness gradient in this second population is lost: individuals from this population are either all equally bad or equally good. In other co-evolutionary methods [20] [16] [17] [10] [6] [9] [12] [5] [23] it has been shown, directly or indirectly, that tests which cause variety in the behaviors of the learners induce a fitness gradient in the learning population. One prime example is the work of Ficici (2001), in which tests are always selected that best distinguish between learners, which in his work are sets of cellular automata rules.

Somewhat surprisingly, it turns out that blindly maximizing the variance in performance among learners is not always helpful, if the learners are trying to approximate the behavior of some external system. Large performance variation may indicate not that there is one or a few learners that do well against this test, but that none of them know how to deal with it properly. For example learners may all react differently to a test, but if that test is then sent to the external system and the output recorded, all of the learners' behaviors may greatly differ from the resulting system behavior. In other words variable behavior before system testing may translate into equally poor performance after testing, thus introducing disengagement. For this reason, simply inducing performance differences in learners will not solve disengagement on its own, if the learners are trying to model an external system; an intermediate level of difficulty must be maintained.

In this paper we introduce such a mechanism, and present it in the context of our co-evolutionary system identification method, the estimation-exploration algorithm (EEA). The new mechanism is referred to as the *test bank*, and the process of mediating test difficulty (which in this case is effected using the test bank) is referred to as *managed challenge*. We illustrate managed challenge by showing how it can be used to accelerate both parametric and symbolic identification of nonlinear systems.

System identification [21] [14] involves the use of a set of input/output data returned by some hidden system to automatically infer the internal structure of that system. System identification approaches are usually divided into four classes: parametric and symbolic identification of linear systems, and parametric and symbolic identification of nonlinear systems. Parametric identification involves the discovery of appropriate parameter values for a target system in which the governing equations are assumed to be known; symbolic (or structural) identification requires the discovery of the governing equations themselves. A host of analytic approaches exist for parametric identification [21] [14], but symbolic identification, especially of nonlinear systems, is beyond traditional approaches.

Evolutionary computation—especially genetic programming—is particularly well suited for symbolic identification because candidate models can be encoded as arbitrarily-sized equations. Several approaches to symbolic identification of nonlinear systems using genetic programming have been attempted [1] [11] [13] [19] [24] [7], but these approaches have focused on how to evolve the equations, rather than how to extract useful information from the hidden system. Chen *et al.* [7] introduced an additive tree encoding that allows for combined parametric and symbolic identification. Winkler *et al.* [24] proposed a hybrid system in which genetic programming is used for symbolic identification and evolution strategies [18] are used for parametric identification. Koza *et al.* [13] proposed a technique in which parametric identification is performed along with symbolic identification directly inside the parse tree.

In many problem domains however, extracting training data from a hidden system can be costly, slow, potentially dangerous[1] and can alter the internal structure of the system itself. For example in medicine it is particularly desirable to perform as few tests as possible, and one way to do this is to form preliminary diagnoses using a few tests, and only then perform additional tests based on the diagnoses.

EEA attempts to automatically produce accurate models of a target system by only performing tests that extract new information from the target system, rather than simply applying a large number of random tests. Heuristic testing is also better than random testing, but this assumes that the algorithm knows something about the structure of the target to be inferred. We use the term *intelligent testing* to indicate that tests should be selected that help us best improve our current models, and that we have little or no *a priori* knowledge about the structure of the target. Bongard *et al.* [3] [2] have shown that this intelligent testing can accelerate identification compared to random testing; we provide further support of this claim here, by performing both parametric and symbolic identification on a nonlinear dynamical system.

However we also show that automated managing of test difficulty—'managed challenge'—can also accelerate system identification. This is done by storing target results which are currently too difficult for the candidate models to explain in a test bank, and withdrawing results when the models are better suited to explaining them. In the context of system identification, managed challenge has two benefits: it keeps disengagement from occurring (i.e. the loss of gradient among the candidate model population); and further reduces the number of target trials required for identification (i.e. difficult tests are not discarded, but are recycled for model refinement at a later time).

In the next section we introduce the estimation-exploration algorithm and the test bank. In section 3 we document the use of this combined algorithm for parametric identification; in section 4 we document the use of it for symbolic identification. In the final section we provide some discussion, concluding remarks, and future research directions.

2. THE ALGORITHM

The estimation-exploration algorithm (EEA) is a coevolutionary algorithm for performing system identification. System identification involves automated construction of a model of a target system, using only input supplied to the system and observed output [14]. The EEA divides the system identification into two components: the generation of accurate models (the estimation phase) and the generation of intelligent input data, or tests (the exploration phase). Bongard *et al.* [3] [2] demonstrated the application of the EEA to a number of system identification tasks, and have demonstrated that by intelligently selecting tests to perform on the target system, the internal structure of the target system can be inferred using less tests than if tests are selected at random.

[1]By definition, the effect of performing a completely new test on an unknown system is more dangerous than performing a similar, but different test to those performed previously.

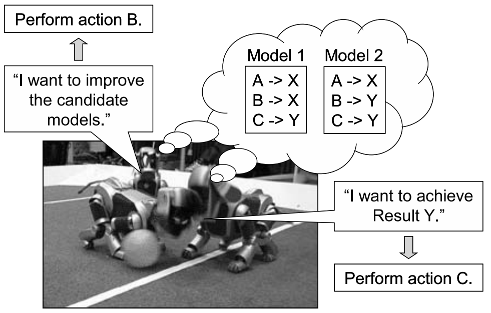

Figure 1: Two complementary definitions of an intelligent test. In this example a RoboCup team has access to a database containing two candidate models of the robot and its environment. The right-hand robot wishes to achieve a particular result, Y, so chooses to perform action C because both models agree that Y will result. The left-hand robot, who wishes to improve the accuracy of the models in the set, chooses to perform action B, because the two models disagree as to the result of this action. Depending on which result occurs, one of the models will be invalidated.

An accurate model is defined as one that produces similar output data as the target system, when both the model and target system are supplied with the same input. An intelligent test can be much more broadly defined. In the case of system identification, an intelligent test is one that indirectly unveils hidden internal components of the target system, thereby allowing the generation of more accurate models. This can be achieved by evolving a test that causes maximal disagreement among the current set of candidate models; the resulting target system output from such a test will provide increased support of some models, and will prove the other models are inaccurate. This way of determining the quality of a test is very indirect, as it does not immediately involve the target system. In other situations, an intelligent test is one that elicits some desirable behavior from the most accurate model currently on hand; if the model is accurate, then that same test should produce the same desirable behavior on the target system. This different interpretation of an intelligent test is shown in Figure 1. In the work presented here the former interpretation of an intelligent test is used: a test is a set of initial conditions that, when supplied to a set of competing candidate models of some target dynamical system, produces the most disagreement between the models.

Figure 2 illustrates the flow of the EEA pictorially. The algorithmic flow of the algorithm is given as follows:

1. **Initialize**
   (a) If an approximate model is available, goto 4.
   (b) If no model is available, generate a random test.
2. **Perform Target Trial**
   (a) Send evolved (or random) test to the target.
   (b) Record the resulting output.

3. **Evolve Candidate Models** (Estimation Phase)
   (a) If this is the first pass through the estimation phase, generate a random set of candidate models.
   (b) If it is the second or subsequent pass, seed the population with the best models evolved so far.
   (c) Provide the evolving models with all previous tests and outputs that have been obtained from the target system, plus the new test/output pair, minus those pairs that are currently stored in the test bank. This set of pairs is known as the *test suite*.
   (d) Output the best candidate models to the exploration phase.

4. **Manage Challenge** (The Test Bank)
   (a) If the most accurate model so far achieved a subjective error[2] below ϵ (i.e. the most recent test was successfully 'digested'), go to (c); otherwise, goto (b).
   (b) Deposit the most recent test in the test bank, along with the target system's behavior in response to this test.
   (c) For each test/output pair in the test bank, add it to the test suite, and compute the current best model's subjective error. If the subjective error for any of these pairs is less than 2ϵ, withdraw it from the test bank, permanently add it to the test suite and goto 3; otherwise, delete it from the test suite and goto 4.

5. **Evolve Informative Tests** (Exploration Phase)
   (a) Always begin the exploration phase with a random population of tests.
   (b) Evolve a test that causes the most disagreement between the candidate model(s) provided by the estimation phase, elicits some desirable behavior from the candidate model(s), or some combination of these two fitness criteria.
   (c) Goto 2).

The EEA is cyclical in the sense that the evolution of models alternates with the evolution of tests, such that test data accumulates over the lifetime of the run: this stands in contrast to many system identification and machine learning methods in which a large amount of training data is gathered before inference begins (eg. [1, 11, 14]). In the EEA, during the nth pass through the estimation phase there are at most n input/output data pairs available for model generation.

The algorithm is evolutionary in the sense that the estimation phase generates a population of candidate models using an evolutionary algorithm, and the exploration phase generates a population of tests also using an evolutionary algorithm. It should be noted however that an alternative search method, a heuristic algorithm or a combination of the two could be used in either phase in lieu of evolutionary search. Co-evolution implies that the evolutionary progress of one population is dependent on the evolutionary progress of the other. In the EEA, tests are evolved using the current most accurate models output by the estimation phase; and models are evolved based on the results of tests evolved in the exploration phase.

[2]Subjective error reports the inaccuracy of the current model only using training data collected from the target system so far.

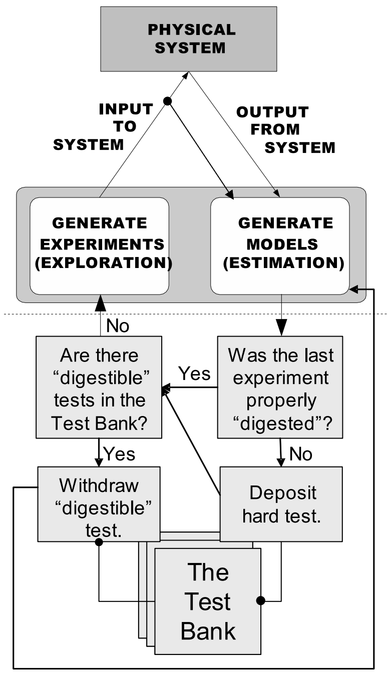
PHYSICAL SYSTEM
INPUT TO SYSTEM
OUTPUT FROM SYSTEM
GENERATE EXPERIMENTS (EXPLORATION)
GENERATE MODELS (ESTIMATION)
No
Are there "digestible" tests in the Test Bank?
Yes
Was the last experiment properly "digested"?
Yes
No
Withdraw "digestible" test.
Deposit hard test.
The Test Bank

Figure 2: The enhanced estimation-exploration algorithm. The original algorithm is shown above the dotted line; the mechanism for 'managing challenge' is shown below the line.

The Test Bank

In initial experiments using EEA to identify nonlinear systems, it was observed that disengagement was occurring. Disengagement has been recognized in other co-evolutionary methods [8] [22], and occurs when the fitness gradient in one population disappears because individuals in that population can no longer successfully respond (however this response is measured) to the set of antagonistic individuals used to determine fitness: i.e., all individuals in the first population perform equally badly when faced with the chosen individuals from the second population. In EEA, disengagement occurs when the algorithm accumulates difficult tests: each successive pass through the estimation phase fails to produce models that can explain those tests (i.e. produce similar behavior as the target system for those particular initial conditions).

In order to combat disengagement, an additional component was added to the original EEA formulation: the test bank. The usage of the test bank is explained in the pseudocode given above. The test bank effectively manages test difficulty; tests that are currently too difficult for the current best model are stored for later use, and tests that have now become easier (they induce an error in the current best model that is equal to or below some threshold) are withdrawn and used to refine the models, instead of performing a new test on the target trial. As will be shown in the results presented in the next two sections, the test bank has two advantages: it prevents disengagement, and it reduces the number of required target trials.

3. PARAMETRIC IDENTIFICATION

In this section we describe the application of the EEA for parametric identification of nonlinear systems. In order to apply the EEA to a novel problem domain, there are six steps that must be followed in order to completely specify the application: characterization of the target system; initialization; estimation phase; exploration phase; termination; and validation.

1) Characterization of the target system.

The target system identified in this approach is a non-linear dynamical system, in which it is assumed that the governing equations are known, but the parameters of the equation are not. In order to test this application we have attempted to identify the two-eyed monster dynamical system, as described in [4], and given as

$$x' = y + y^\alpha \quad (1)$$
$$y' = \delta x + \frac{\eta}{\beta} y - xy + \frac{\gamma}{\beta} y^\alpha, \quad (2)$$

where $[\alpha, \beta\, \gamma, \delta, \eta] = [2, 6, 5, -1, 1]$ are the hidden parameters of the target system, and correspond to the specific instance of the two-eyed monster system described in [4]. Given some initial conditions $IC = [x_0, y_0]$, the target system is integrated using the Runge-Kutta method, using a step size of $h = 0.01$, from $t_0 = 0s$ to $t_{\text{end}} = 2s$.

2) Initialization. To initiate the EEA, a random initial condition for each variable is chosen from the range $[-10, 10]$, and supplied to the target system. The resulting time series of x and y are returned to the estimation phase.

3) Estimation Phase. Each genome in the estimation phase encodes a candidate model of the target system. For parametric identification, the genotype is the string of five unknown parameters $[\alpha, \beta\, \gamma, \delta, \eta]$, encoded as real values. At the beginning of the first pass through the estimation phase, 300 random genomes are generated: genome values can range between $[-10, 10]$.

Evaluation of a genome proceeds as follows. The generic target equations are labeled using the five encoded parameters specified in that genome. The resulting model is presented with all of the initial conditions currently in the test suite; for each test, the model is integrated using the Runge-Kutta method with $h = 0.01$ from $t_0 = 0s$ to $t_{\text{end}} = 2s$, and the resulting time series are recorded. Once all tests have been run on the current model, the model's subjective error is computed using

$$s_e = \frac{\sum_{i=1}^{v} \sum_{j=1}^{n} \max(|t_{ij}^{(1)} - m_{ij}^{(1)}|, \ldots, |t_{ij}^{(n)} - m_{ij}^{(n)}|)}{vn}, \quad (3)$$

where $v = 2$ is the number of state variables in the system; n is the number of tests currently in the test suite; $t_{ij}^{(k)}$ is the value of variable i, at time interval k, produced by the target system using test j; and $m_{ij}^{(k)}$ is the value of variable i, at time interval k, produced by the candidate model using test j. This metric gives an approximate measure of how well the current model matches the observed behavior of the target system. The *max* term ensures that the system will first generate models that mimic major transients in the target system, such as narrow and tall spikes in the time series, because such transients produce large differences over short time periods.

Once all of the genomes have been evaluated, deterministic crowding [15] is used to produce a new generation of genomes. All genomes are grouped into pairs at random, and are then crossed using one-point crossover, and then mutated. Mutation is as follows. A single value per genome is chosen, and is either: replaced with a new random value taken from $[-10, 10]$ (probability= 0.5); increased by 10^κ (p= 0.25), where κ is a random real number taken from $[-7, 1]$, or decreased by 10^κ (p=0.25)[3]. The two resulting child genomes are compared against both parent genomes using variational distance, and are paired with the most similar parent. The new child genomes are evaluated, and if a child achieves a lower subjective error than its parent, the child replaces the parent. This process is continued until a model achieves a subjective error of $s_e \leq \epsilon (= 0.1)$, or until the most accurate model in the population has not been replaced for 20 generations. We have yet to investigate systematic methods for determining ϵ given some target system, or how sensitive the algorithm is to this parameter. Additional investigation into the sensitivity of all of the parameters required by the algorithm is warranted.

When the estimation phase terminates, the 10 best models are output. On the second and subsequent passes through the estimation phase, the most recent 10 best models are used to seed the initial random population.

4) Exploration Phase. If a test is not withdrawn from the test bank, then a new one must be created. This is accomplished in the exploration phase. Each pass through the exploration phase begins with an initial random population of 300 genomes (tests from previous passes are not used to seed the initial random population[4]). Each genome is composed of v real values (in this case $v = 2$), where each value indicates the initial value of the corresponding variable. Each genome is evaluated as follows. The encoded initial condition is supplied to each of the 10 models output by the estimation phase, and the resulting behaviors are recorded. The fitness of a test is then given as

$$t_f = \frac{\sum_{i=1}^{v} \sigma(m_{i1}^{(n)}, \ldots, m_{i10}^{(n)})}{v}, \quad (4)$$

where m_{ij}^n is the final value of the ith variable from model j, and $\sigma(m_{i1}^{(n)}, \ldots, m_{i10}^{(n)})$ gives the variance of these final signals across all 10 models.

When all of the tests have been evaluated, a new generation of genomes is produced using deterministic crowding as explained in the previous sub-section. In this case however, child genomes that induce higher model variance than their assigned parent genome replace it. The exploration phase is continued for 30 generations, and the test with the highest fitness is sent to the target system for evaluation.

5) Termination. Termination occurs when the algorithm has passed through the estimation phase 40 times. In cases when the test bank is used, there may be fewer than 40 passes through the exploration phase, and fewer than 40 target trials.

6) Validation. After each pass through the estimation phase, the best model was saved, and its objective error was calculated using

$$o_e = \frac{\sum_{i=1}^{v} \sum_{j=1}^{1000} \max(|t_{ij}^{(1)} - m_{ij}^{(1)}|, \ldots, |t_{ij}^{(n)} - m_{ij}^{(n)}|)}{vt}, \quad (5)$$

where 1000 random, unseen initial conditions are generated as test data.

Parametric Identification Results

Four sets of 60 independent runs were performed against the target system. In the first set, both intelligent testing and the test bank were disabled: the exploration phase simply outputs a random test instead of evolving one, and tests are never stored or retrieved from the test bank. In the second set, the exploration phase is enabled and the test bank is disabled. In the third set, the exploration phase is disabled and the test bank is enabled. In the fourth set, both the exploration phase and the test bank are enabled.

Figure 3 reports the mean behaviors of these four algorithm variants. Figure 3a reports the mean objective errors

[3]This mutation operator seemed to do well for our problem; however more standard operators such as the Gaussian or adaptive Gaussian operator could also be used.

[4]Although this might further empower the algorithm.

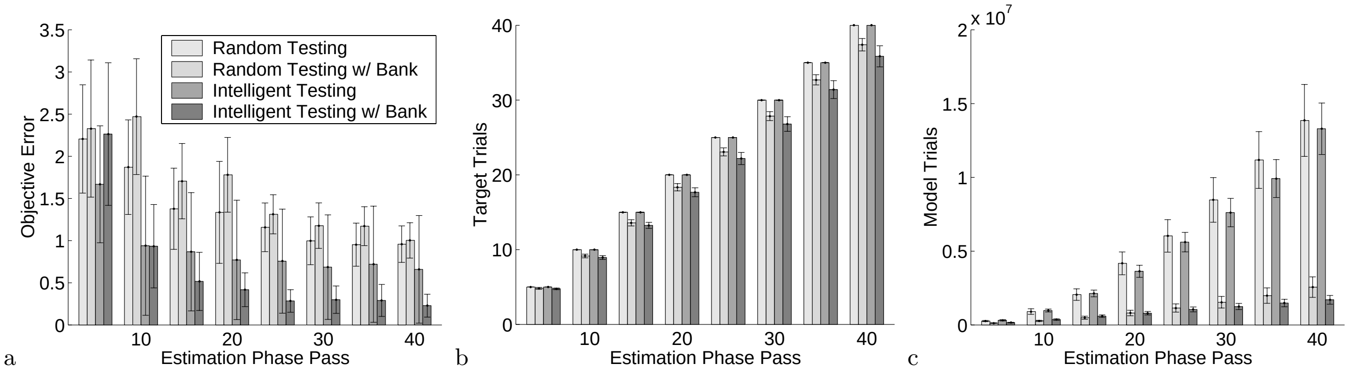

Figure 3: Mean performance of the four algorithm variants for *parametric* identification. a: Mean objective errors of the best models produced after each pass through the estimation phase. b: The mean number of target trials performed, as a function of the number of elapsed passes through the estimation phase. c: The mean number of model evaluations performed, as a function of the number of elapsed passes through the estimation phase. All error bars in this paper report standard error.

of the best models produced by each pass through the estimation phase. In some cases the objective error of a model approaches infinity, due to some inaccuracy of the model that causes positive or negative feedback. In these cases the model is discarded and the mean objective is calculated using only the remaining well behaved models[5]. Figures 3b and 3c report the mean number of target trials and model evaluations that have been conducted so far, at the end of that particular pass through the estimation phase, respectively.

Figure 3a indicates that both the use of intelligent testing, as well as test difficulty mediation via the test bank, improves the ability of the algorithm to discover accurate models. If either intelligent testing or test difficulty mediation is disabled (or both), the algorithm discovers, on average, less accurate models. The one exception is the statistical overlap between the third and fourth variants: intelligent testing without the test bank exhibits much more variable performance, and since its standard error overlaps with intelligent testing with the bank, which has much more consistent behavior.

Furthermore, Figure 3b indicates that the fourth variant is able to discover more accurate models using less target trials than the other three variants. This indicates that the test bank not only helps to manage test difficulty, but also statistically significantly reduces the number of target trials required per run of the algorithm, compared to runs that do not use the test bank: without the test bank, the number of target trials required is equal to the number of passes through the estimation phase; with the test bank, fewer trials are required. Finally, Figure 3c indicates that using the test bank significantly reduces the number of model evaluations performed during a run. This is because when the test bank is not used, each model must be evaluated i times, using the i tests generated so far, during the ith pass through the estimation phase; when using the test bank, some tests may be stored in the bank, and so that there may be less than i tests in the test suite during the ith pass through the estimation, thereby reducing the total number of required model evaluations.

4. SYMBOLIC IDENTIFICATION

The EEA has also been used for symbolic identification in nonlinear systems, in which it is assumed that the equations governing the target system are unknown.

[5]A model may be well behaved but still inaccurate

1) Characterization of the target system. The target system is the same as that used in the previous section, the two-eyed monster. It is assumed that the equations given in Eqn. 2 are unknown. However, as has been pointed out in [7], it is difficult to identify both the structure and parameters of a hidden system simultaneously, so we have assumed that the five parameters $[\alpha, \beta\,\gamma, \delta, \eta] = [2, 6, 5, -1, 1]$ are known, and do not have to be identified. We are currently devising a hybrid evolutionary approach that allows us to evolve both the structure and parameters of the hidden system simultaneously.

2) Initialization. Initialization is the same as for parametric identification: a random set of initial conditions is drawn from $[-10, 10]$, supplied to the target system, and the resulting state variable time series are returned, along with the test, to the EEA.

3) Estimation Phase. In the estimation phase, each genome encodes a set of differential equations that, when labeled with the five known parameters, produces a candidate model. Each genome is encoded as a forest in which there is one parse tree for each state variable, and each parse tree encodes the differential equation for that variable. Nonterminal nodes are labeled from the set $[\sin, \cos, +, -, *, \%,$ pow], where % is protected division, and terminal nodes are labeled from the set $[v, p, t]$, where v indicates a state variable, p indicates a parameter, and t represents the current time in seconds. Terminal nodes also have an associated real value that is rounded to an integer in $[0, 1]$ when the terminal node has label v; to an integer in $[0, 4]$ when the terminal node has label p; and is discarded for label t. The integer is treated as an index when paired with the terminal label: for example $v(0) = x$, $v(1) = y$, $p(0) = \alpha$, and so on. The maximum depth allowed for any tree was set to 5, which is the maximum depth of the y' tree when the target system is cast as a parse tree. Other symbolic encodings are possible, and may indeed further improve the performance of this algorithm. However the focus of this paper is not to demonstrate a superior symbolic encoding, but to demonstrate the usefulness of combined intelligent testing and test difficulty mediation.

Thus the domain knowledge that we bring to this problem is as follows: the number of state variables; the number and value of the parameters; the required operator types; and the general size of the expected solution.

During the first pass through the estimation phase, 300 random forests are generated. For each node that is created in each tree, a label is chosen from the combined set $[\sin, \cos, +, -, *, \%, \text{pow}, v, p, t]$ with equiprobability if the de-

Table 1: Symbolic Representations of Three Successive Models

	x’	y’
Target	$\mathbf{y + y^2}$	$\mathbf{x + \frac{1}{5}y - xy + \frac{6}{5}y^2}$
Best After First Pass	$2t - \cos(5) - 1 - x - ty + \mathbf{y^2}$	$(\cos(t/y) + 6)^{-x} - 5y$
Best After Second Pass	$\mathbf{y} - \sin(2) + \mathbf{y^2}$	$\sin(\cos(1)) + 1 + y - \mathbf{xy} + y^2$
Best After 40th Pass	$\mathbf{y + y^2}$	$((((\sin(t-t)) - ((t-y) * y)) + x) - ((((p4 - y)/p2) + (x - t)) * y))$ $= -ty + y^2 - x - (-\frac{y}{5} - \frac{y^2}{5} + xy - ty)$ $= \mathbf{-x + \frac{1}{5}y - xy + \frac{6}{5}y^2}$

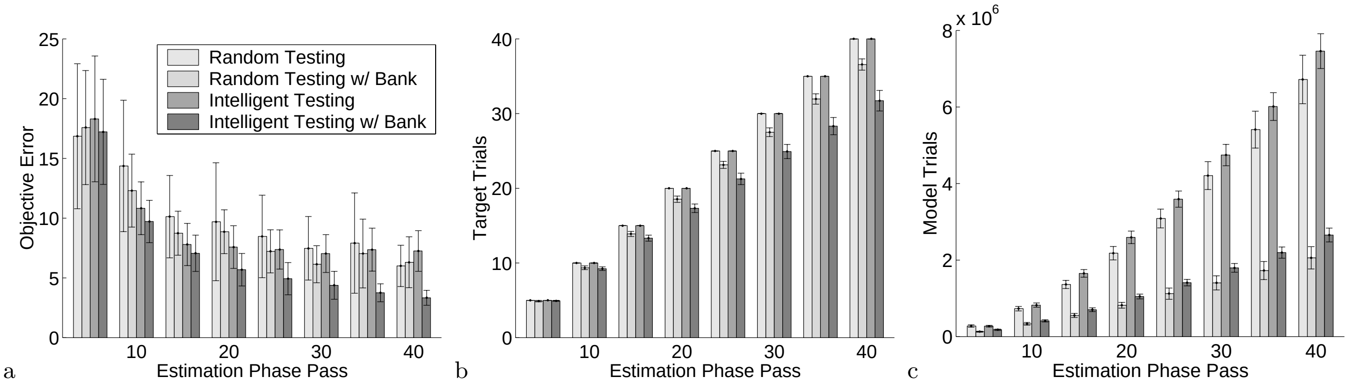

Figure 4: Mean performance of the four algorithm variants for *symbolic* identification. a: Mean objective errors of the best models produced after each pass through the estimation phase. b: The mean number of target trials performed, as a function of the number of elapsed passes through the estimation phase. c: The mean number of model evaluations performed, as a function of the number of elapsed passes through the estimation phase.

pth is less than five, and from the terminal set $[v, p, t]$ with equiprobability otherwise. A random real number is also selected for the node. If the arity of the labeled node is one or more, a left subtree is created; if the arity is two, a right subtree is also created.

Each forest is evaluated as follows. For each test the current state variable values, the parameter values and t are supplied to the forest, and x' and y' are calculated. These values are used by Runge-Kutta integration to produce new x and y values. The model is evaluated for the same time period as the target system (2s). The subjective error of the current genome is then calculated using Eqn. 3.

Once all genomes have been evaluated, deterministic crowding is again used to produce a new generation of genomes. Mutation is as follows: For each tree, a node is selected at random. If the node is a terminal, then the associated real value may be replaced with a new random value in $[0, 1]$ (p= 0.5), or the node may be mutated (p= 0.5). If the selected node is non-terminal, it is always mutated. Node mutation involves the selection of a new label from the combined set if the depth is less than five or from the terminal set if the depth is equal to five. If the new label has an arity different from the original node label, then new subtrees may have to be created (using the process of node creation described above) or deleted. Crossover is also used, in which one node from each of the parent forests may be selected, and subtree crossover carried out (node selection is not restricted to the same state variable tree in both parent forests). The child forests are then evaluated, and those with lower subjective errors than their associated parents replace them. Similarity between child and parent tree is taken to be the absolute difference between their size (number of nodes).

Evolution continues until a forest is discovered with $s_e \leq \epsilon (= 1)$, or until the best model has not been replaced for 20 generations[6]. If at the termination of the pass the best model has $s_e > 1$, then the most recent test is stored in the bank; otherwise the test is added to the test suite. If there is a test in the bank that, when added to the test suite induces an error of $s_e \leq 2$, it is withdrawn and added to the test suite, and the next pass in the estimation phase commences. Otherwise, the exploration phase commences using the 10 best models output by the current pass through the estimation phase. On the second and subsequent passes through the estimation phase, the 10 best models from the previous pass seed the initial random population.

4) Exploration Phase. The exploration phase is conducted identically as described for parametric identification.

5) Termination. Termination occurs when 40 passes have been made through the estimation phase.

6) Validation. The objective error of the best model produced by each pass through the estimation phase is calculated using Eqn. 5.

Symbolic Identification Results

As for parametric identification, four algorithm variants were formulated, and 60 independent runs were conducted for each variant. Figure 5 illustrates the behavior of three models taken from a typical run of the fourth algorithm variant, in which both intelligent testing and the test bank are used: the best model after the first pass through the estimation phase, the best model after the second pass, and the best

[6] ϵ is greater than it was for parametric identification because candidate models have, on average, higher s_e in this case than for parametric identification. If ϵ is too low, then all tests are placed in the test bank and little identification occurs.

model after the 40th pass. Each model was exposed to four different initial conditions, and the behavior of the target system is compared against the behavior of that model. Table 1 reports the equations encoded by these models. The reported equations are compressed but identical versions of the actual encoded equations; the compression of one of the encoded equations is shown (compression was done manually). As both the figure and table indicate, by the 40th pass an exact model was found. Also, the second model has captured some of the transients in the target system (Figure 5, middle panel), even though those transients occur after the time period for which the models were evaluated. The abilities of the models to mimic the behavior of the target system beyond the time period for which they were evaluated indicates that the models are not just matching training data, but are successfully generalizing based on the training data that has been collected. This ability to generalize has been seen during the identification of other target systems (results not shown). The good generalization is attributed to the fact that the algorithm has the chance to find tests that 'break' early overspecialization in evolved models.

Figure 4 reports the mean identification ability of the algorithm variants for the given nonlinear target system. Figure 4a indicates that both intelligent testing and test difficulty mediation significantly improve the algorithm's ability to perform symbolic identification. This is supported by the fact that the mean performances of the three algorithm variants that lack either intelligent testing, test difficulty mediation produce significantly less accurate models, on average. Figure 4b indicates that the use of these two mechanisms allows the algorithm to produce more accurate models with fewer target trials, compared to the other three algorithm variants. Figure 4 also indicates that the test bank reduces the total number of model evaluations that are performed during a run, except for random testing with the bank. However, random testing with the test bank still produces much more inaccurate models than intelligent testing with the test bank.

Figure 4b indicates that intelligent testing, together with the test bank, requires less target trials than the algorithm variant with random testing and the test bank. The reason for this is explained by Figure 6, which reports the mean number of tests stored in the test bank after each pass through the estimation phase for these two algorithm variants. Clearly, the intelligent testing variant accumulates many more tests in the bank than the variant with random testing does. This is explained by the fact that random testing produces, on average, 'easy' tests that the current approximate models can explain such that they achieve a subjective error below 1, and therefore the test becomes a permanent member of the test suite, even though the objective error of the model is relatively high. However difficult tests, such as those that produce drastic value changes during integration, are rarely discovered through random testing, although they are discovered much more often using intelligent testing.

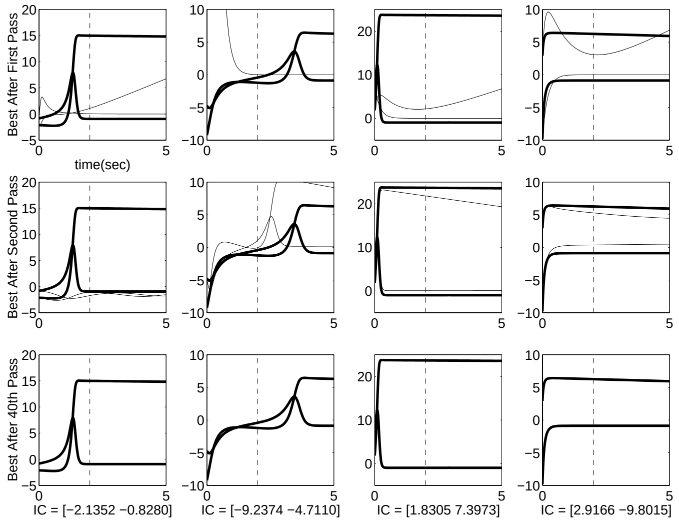

Figure 5: The behaviors of three successive models from a typical run, and the target model. The four columns correspond to four different tests (initial conditions indicated by IC $= [x0, y0]$). The three rows correspond to the best model output by the first, second and 40th pass through the estimation phase, respectively. Thick lines indicate the behavior of the target system; thin lines indicate the behavior of the model. The dotted line indicates the time period for which the target and models were evaluated; the behaviors of the target and models were continued for another 3s to indicate the generality of the models.

5. DISCUSSION AND CONCLUSIONS

In this paper we have described the use of a co-evolutionary algorithm, the estimation-exploration algorithm, for use in parametric and symbolic identification of nonlinear dynamical systems. The co-evolution derives from the antagonistic interplay between an evolving population of candidate models, and an evolving population of tests: test fitness is defined in terms of how much variance they induce in the candidate models, and model fitness is defined in terms of how closely they can match the observed behavior of the target system when supplied with a highly fit test. Tests that induce a lot of model variance, and are then supplied to the target system, often produce target behavior that is difficult to explain using the current set of candidate models. If this happens too often, disengagement occurs: it becomes difficult to determine the relative fitness of models because they all perform equally badly on the set of difficult tests.

In response to this difficulty we have here introduced a new component to the estimation-exploration algorithm, the test bank. The test bank is one way to 'manage challenge' in co-evolutionary systems. We have here introduced the term 'managed challenge' to refer to any process that automatically and dynamically mediates teaching difficulty in a co-evolutionary system in order to diminish the threat of disengagement. Managed challenge is effected by the test bank in the following way: tests that are currently too difficult to explain by the current best models are stored in the bank (along with the target's behavior in response to this test), and those in the bank that are now close to being explained by the current best models are withdrawn and added to the current training set. This contrasts with other co-evolutionary methods, of which Ficici (2001) is a prime example, in which there is no static target system which can be used to determine whether induced disagreement is reducing disengagement, or exacerbating it.

We have shown that for both parametric and symbolic identification, the use of the test bank has two advantages: first, it allows for the discovery of more accurate models; and second, it allows for fewer target trials to be performed because in some cases a test is withdrawn and re-explained, rather than generating a new test to perform on the target system. In many system identification problem domains it is costly, slow, dangerous and alters the state of the system if many target trials are performed, so the usual metric for judging the quality of system identification methods is twofold: how accurate a model can it produce, using as few target trials as possible.

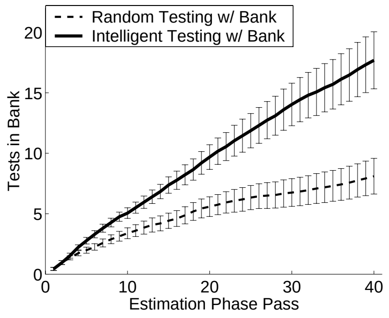

Figure 6: The number of tests stored in the test bank, as a function of the number of elapsed passes through the estimation phase, during symbolic identification.

Our method therefore improves on other evolutionary approaches to system identification [1] [11] [13] [19] [7], in which it is assumed that training data is generated at random. The results presented here support the claim made by Bongard *et al.* [3] [2] that intelligent testing is better than random testing because it leads to more accurate models, on average, using the same amount of training data. However the danger of intelligent testing is that it generates too difficult training data that is hard to explain at the outset; the test bank mitigates this effect by allowing for the dynamic storage and retrieval of training data. In short, we have shown that both intelligent testing, along with 'managed challenge', improves model accuracy, and does so using fewer target trials.

We are currently formulating a genetic encoding that allows simultaneous parametric and symbolic identification. We are also exploring the use of the test bank and other 'managed challenge' mechanisms to combat disengagement in other co-evolutionary systems. We are also integrating the method with automated biological experimentation systems such that evolved tests can be carried out on biological organisms in order to infer their genetic regulatory networks.

6. REFERENCES

[1] H.W. Andrew. System identification using genetic programming. In *Proceedings of the Second Intl. Conf. on Adaptive Computing in Engineering Design and Control*, pages 57–62, 1996.

[2] J. C. Bongard and H. Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In *Proceedings of The 2004 NASA/DoD Conference on Evolvable Hardware*, pages 169–176, Seattle, WA, 2004.

[3] J. C. Bongard and H. Lipson. Automating genetic network inference with minimal physical experimentation using coevolution. In *Proceedings of The 2004 Genetic and Evolutionary Computation Conference*, Seattle, WA, 2004.

[4] R.L. Borrelli and C.S. Coleman. *Differential Equations: A Modeling Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] A. Bucci, J.B. Pollack, and E.D. De Jong. Automated extraction of problem structure. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 501–512. Springer-Verlag, 2004.

[6] J. Cartlidge and S. Bullock. Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. In *Congress on Evolutionary Computation*, pages 1420–1425, Piscataway, NJ, 2002. IEEE Press.

[7] Y. Chen, J. Yang, Y. Zhang, and J. Dong. Evolving additive tree models for system identification. *International Journal of Computational Cognition*, 3(2):19–26, 2005.

[8] D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *European Conference on Artificial Life*, pages 200–218. Springer Verlag, 1995.

[9] B. Dolin, F.H. Bennett III, and E.G. Rieffel. Co-evolving an effective fitness sample: Experiments in symbolic regression and distributed robot control. In *ACM Symposium on Applied Computing, SAC 2002*, pages 553–559, 2002.

[10] S.G. Ficici and J.B. Pollack. Pareto optimality in coevolutionary learning. In *Advances in Artificial Life: 6th European Conference (ECAL 2001)*, pages 316–327. Springer Verlag, 2001.

[11] G.J. Gray, D.J. Murray-Smith, Y. Li, K.C. Sharman, and T. Weinbrenner. Nonlinear model structure identification using genetic programming. *Control Engineering Practice*, 6:1341–1352, 1998.

[12] E.D. De Jong and J.B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192, 2004.

[13] J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1999.

[14] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1999.

[15] S.W. Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving from Nature – PPSN-1992*, pages 27–36, 1992.

[16] B. Olsson. *NK*-landscapes as test functions for evaluation of host-parasite algorithms. In *Parallel Problem Solving from Nature – PPSN-2000*, pages 487–496, 2000.

[17] J. Paredis. Towards balanced coevolution. In *Parallel Problem Solving from Nature – PPSN-2000*, pages 497–506, 2000.

[18] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer System nach Prinzipien der biologischen Evolution*. Fromann-Holzboog, Stuttgart, DE, 1994.

[19] K. Rodriguez-Vazquez and P.J. Fleming. Genetic programming for dynamic chaotic systems modelling. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, pages 22–28, 1999.

[20] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[21] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[22] R.A. Watson and J.B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector and E.D. Goodman *et al*, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 702–709, San Francisco, CA, 2001. Morgan Kaufmann.

[23] R. P. Wiegand and J. Sarma. Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In *Parallel Problem Solving from Nature – PPSN-2004*, page 912921. Springer, 2004.

[24] S. Winkler, M. Affenzeller, and S. Wagner. New methods for the identification of nonlinear model structures based upon genetic programming techniques. In *Proceedings of the 15th International Conference on Systems Science*, volume 1, pages 386–393, 2004.