

Co-Evolutionary Methods in System Design and Analysis

H. Lipson, J. Bongard, V. Zykov

Computational Synthesis Laboratory,

School of Mechanical & Aerospace Engineering, Cornell University, Ithaca NY 14853, USA

hod.lipson@cornell.edu

Abstract

Co-evolution is a biological process where populations of interacting individuals challenge each other in an ongoing cycle of adaptation, such as predator-prey competition and symbiotic cooperation. Though co-evolution can potentially drive progress beyond stagnation point of conventional evolutionary algorithms, its application in practice has been challenging due to its complex dynamics. Here we show a systematic approach for implementing co-evolution in both the design and analysis of physical systems. The implementation is based on co-evolving predictive models and design steps. A number of applications in robotics and regulation networks are shown.

Keywords:

Co-evolutionary design, design automation, computational synthesis

1 INTRODUCTION

Co-evolution is a biological process where populations of interacting individuals challenge each other in an ongoing cycle of adaptation [26]. Co-evolution can take many forms: The antagonistic arms-races of a predator and prey, the symbiotic cooperation between individuals in an ecosystem, or the mutual exploitation of a host and a parasite. Though co-evolution is often cited as one of the driving forces behind continuous innovation in nature, its complex dynamics have prevented it from being applied systematically as a computational metaphor. Despite few notable successes, its application to practical problems is challenging: Co-evolution often results in collusion, cycling, and disengagement. Nevertheless, as the limitations of conventional evolutionary computation models are being identified [16], understanding the continuous drive for improvement potentially offered by co-evolution and finding systematic ways to its application become crucial to progress in this field. This paper introduces a systematic, domain independent method for performing synthesis or analysis using coevolution.

2 BACKGROUND

2.1 Co-evolution as a metaphor of intelligent reasoning

Though the importance of co-evolutionary processes is well established in nature, they are perhaps equally useful as metaphors of intelligent processes in other domains. It is

revealing to watch how a scientist, for example, progressively uncovers a new theory: The scientist holds several alternative hypotheses that might explain empirical observations, and also holds several candidate experiments that could be carried out to distinguish among those potential hypotheses. The candidate experiment that makes the current candidate hypotheses disagree the most in their predictions is the experiment that is most worth carrying out, as data collected from it will refute some hypotheses and improve others. Like a predator and prey, this “co-evolution” of hypotheses and tests allows tests to continuously target weaknesses (disagreement) in hypotheses, driving them forward and eliminating cases of both over-generalization and over-fitting. In this case, co-evolution is used for analysis.

In a second example, an engineer is progressively designing a system. The engineer understands the system being designed in several different “abstractions”, and also contemplates several possible “actions” that could be done to improve the design. The action that various abstractions – though different – all agree that would result in improvement, is the action that is most worth carrying out as it circumvents the uncertainties in his understandings. Like symbionts in an ecosystem, this “co-evolution” of *actions* and *abstractions* allows continuous improvement despite weaknesses in understanding. In this case, co-evolution is used for synthesis.

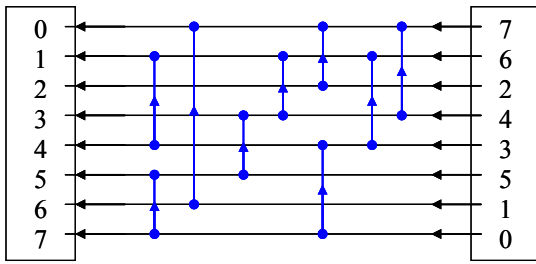


Figure 1. A sorting network accepts an arbitrary sequence of numbers on the right, and through a set of fixed comparators (shown as arrows), guarantees that the sequence are output sorted on the left.

In many cases, however, analysis and synthesis are coupled. We need create models of systems in order to meaningfully synthesize or control them, and we need to control them meaningfully (e.g. carry out experiments) in order to understand and create models of them. The three examples above – co-evolution for synthesis, for analysis, and for both simultaneously, constitute the key questions our work seeks to answer.

2.2 Co-evolutionary computation

There has been growing interest in coevolutionary algorithms within the evolutionary computation community, as evidenced by the recent literature (eg. [29,28,24,14,11,18]), starting with the seminal work of Hillis [17] on sorting networks. Contrary to conventional evolutionary systems, coevolutionary systems consist of one or more populations where individuals may influence the relative ranking of others individuals [11]. For example, whether individual A is inferior or superior to individual B may depend on a third individual C rather than on some external fitness metric that provides an absolute ranking. There are a number of different forms of co-evolution: Antagonistic coevolution (e.g. predator-prey), cooperative coevolution (e.g. symbiosis [33]) or non-symmetric systems (e.g. host-parasite or teacher-learner [15]).

In the discussion of coevolutionary systems it is important to distinguish between the notion of *objective* fitness versus *subjective* fitness. Objective fitness is the well-defined absolute fitness metric used in a classical evolutionary algorithm. Subjective fitness is the fitness as measured by coevolving individuals, and may be only weakly correlated with the true objective fitness, and may sometimes even be misleading. A coevolving individual only knows its subjective fitness. In the examples presented in this paper we use absolute fitness only for benchmarking purposes; the algorithm itself has no access to the absolute fitness.

An example

One of the classical examples of a successful application of co-evolution to solve a computational problem is the work of Hillis [17]. In that work, evolutionary algorithms were used to design sorting networks: These are networks that accept arbitrary sequences of numbers and through a set of fixed comparators (Figure 1), guarantee that the sequences are sorted. Design of such sorting networks poses a significant challenge and was a subject of many papers. Initially, Hillis

used a conventional genetic algorithm to evolve network designs. The network's fitnesses were determined by sampling a large number of sets of randomly-ordered sequences, and counting the fraction that was sorted correctly. As in many evolutionary computation examples, the process initially made progress, but stagnated after a while yielding less than optimal networks.

In the following set of experiments, Hillis allowed the test sequences themselves to co-evolve with the networks. While the fitness of the networks remained the fraction of sequences they sorted correctly, the fitness of sequences was the fraction of networks they could defeat. This co-evolutionary setup drove the system to high performance, yielding designs that outperformed some previously published results.

2.3 Challenges in co-evolutionary computation

Since this seminal success, however, subsequent implementations of co-evolution ran into a variety of unanticipated difficulties. Implementations of coevolution have been plagued with a number of pathologies arising from their complex coevolutionary dynamics [32]. Much of the focus of current research has been to address these drawbacks [29,14,30].

1. **The “red queen” effect** is a consequence of the purely subjective measure of fitness in coevolutionary systems: Two populations continuously adapt to each other and their subjective fitness improves, but they fail to make any consistent progress along the objective metric [13]. Conversely, they do make progress along an objective fitness but their subjective fitness does not reflect this and falsely indicates lack of progress.
2. **Disengagement between populations.** In a co-evolutionary setup, one population provides a fitness gradient for the other population to follow. If one population far exceeds the performance of the other population, then within each population all individuals appear to have equal subjective fitness. This loss of gradient precludes any further progress on both sides.
3. **Cycling, defocusing, and transitive dominance.** Because fitness measure is co-evolving, a population may at some point “forget” what it has previously known because it is no longer part of the new fitness criteria. This effect often leads to cycles, where individuals improve one weakness at the expense of another weakness, known as “defocusing”. This multidimensionality of the fitness landscape may also results in transitive dominance among individuals.

3 THE EXPLORATION-ESTIMATION ALGORITHM

3.1 Overview

We have developed an inference algorithm that can be used to intelligently formulate actions the system can execute that either elucidate uncertainties about the world or affect it

despite uncertainty [21,7]. Like the game of “20 Questions”, (determine the object I’m thinking about using up to 20 yes/no questions), the algorithm progressively constructs a model of the hidden system by carefully formulating new intelligent questions based on answers to previous ones. Our current implementations suggest that the algorithm can be applied to a variety of problem domains: To date the algorithm has been used to learn deterministic finite automata [3], commonly referred to as grammar induction, an important problem in machine learning; for reconstructing gene regulation networks using gene expression data [6], an important problem in system biology; and the evolution of robot controllers given some desired task [2,4,5,8], an important problem in robotics research.

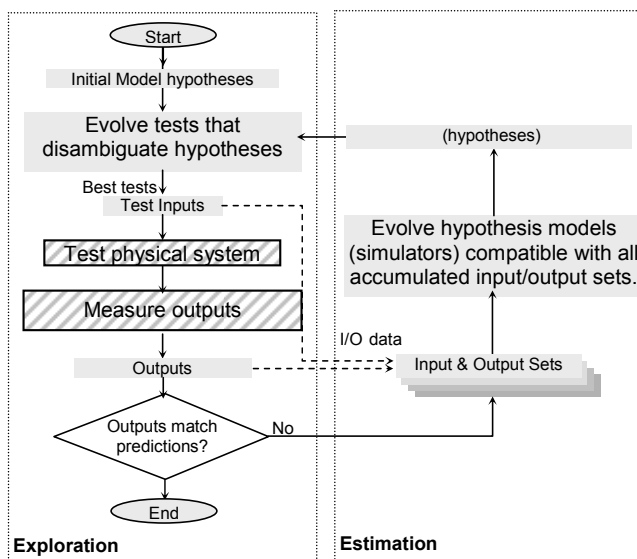


Figure 2. The exploration-estimation algorithm for actively creating models of the world and actions in it.

3.2 Algorithm outline

The estimation-exploration algorithm is a parallel algorithm that simultaneously manipulates multiple candidate system models and multiple candidate actions (Figure 2). The method by which candidate world models and candidate actions evolve over time is as follows. World models are rewarded for being able to correctly explain all the input-output data collected so far from interaction with the real world. Multiple candidate models exist because there may be many ways to explain observed input-output data, and because there is uncertainty about some observations. Candidate actions are rewarded for either creating disagreement among model predictions (thereby elucidating uncertainty), or by exploiting agreement (thereby capitalizing on certainty).

The algorithm implementation iterates through a two-phase exploration-estimation cycle, as shown in Fig. 2. The algorithm starts with a set of different, possible models, which represent the user’s best guesses as to the internal structure of the target system. A population of candidate actions is then evolved to create maximal disagreement (or variance) among the predictions of the current models, or to achieve a desired goal. The best action(s) are then performed on the system, after which the estimation phase

then evolves multiple internal models so as to explain observed input-output data. The process then iterates: The exploration phase seeks new actions using the improved models, and so forth, until one of several termination criteria are met (see below).

Before starting, the following components are required:

- A representation, search operators and dissimilarity metric for worlds
- A representation and search operators for inputs
- A representation and dissimilarity metric for outputs

1. Initialization:

- a. Define a set of initial models (hypotheses) of the system. They can be blank, random, or seeded with some prior knowledge.

2. Exploration phase

- a. To learn: search for an input that creates the most variation among the outputs of the current set of model hypotheses.
- b. To achieve a goal: search for an input that creates the desired output with least variation among the current set of model hypotheses.
- c. Carry out the best test on the target system by applying that input and measuring the outputs. This input-output set is stored.

3. Estimation phase

- a. Search the model space for models that are maximally consistent with *all* stored input-output sets. Encourage diversity [23].

4. Termination

- a. The exploration-estimation cycles are repeated until one of these conditions is met:
 - i. The desired output has been achieved.
 - ii. The estimation phase is unable to produce a model consistent with all the accumulated observations. In this case the algorithm fails, but avoids blind trials (but see section 4.5).
- b. The estimation phase repeatedly produces the same set of model hypotheses for a number of cycles in a row, or the exploration phase is unable to find an input that causes variation in the outputs of the current model hypotheses. The hypotheses may be identical, indicating there is one solution; or different, indicating that some aspect of the system is unobservable and several solutions are possible but cannot be disambiguated by the inputs.

5. Validation

Cross-validate the correctness of the model using unseen data.

3.3 Modes of failure or non-convergence

If the algorithm fails, there are a number of possible causes:

- The *physical* test was not carried out properly (the inputs were not set correctly or the measurements of the outputs were incorrect).
- The search process in the estimation phase is not able to find the best models (the actual cause depends on the search technique used, e.g. could be stuck in local minima, or tests are too difficult – see section 4.5)
- The system or necessary inputs are outside the space spanned by the representation language – the representation is not sufficiently general.
- The system behaves inconsistently. In such cases the space of systems must be broadened to include additional, possibly non-deterministic or time-dependent elements.
- There are several different models that describe the system equally well.

If the algorithm continues running but does seem to make progress (i.e., is slow to converge), there are a number of possible causes:

- The search process in the estimation phase does not produce a diverse enough set of hypotheses (lack of diversity). In this case the inputs generated in the exploration stage do not produce much information beyond what is already known and the algorithm cannot make progress.
- The search process in the exploration phase is not able to find the best inputs (the actual cause depends on the search technique used).

4 APPLICATIONS

4.1 Inference of a state machine

We have implemented the exploration-estimation algorithm for the problem of uncovering the hidden structure of hard-to-observe (i.e., unbalanced) finite state machines (FSMs). The problem of grammar induction is widely used as a benchmark of inference algorithms [12,27,19]. FSMs are legendarily difficult to infer because they take an input string (made of ones and zeroes in our case) that produce complex changes in internal states to produce a single output (a one or a zero); unbalanced FSMs are even more elusive because they favor one output classification over the other. FSMs are inferred by feeding test input strings and trying to predict their output correctly. Maximal information extraction is achieved when the numbers of negative and positive classifications are comparable, indicating that the input strings are probing the boundary of the model. In a set of 3000 experiments over 30 different hidden FSM networks of various levels of balance, our exploration-estimation algorithm showed the ability to create predictive models of the hidden system, and do so with significantly fewer tests as compared to a control algorithm that employed random trials.

Figure 3 shows the resulting classifications for 30 independent runs of the proposed algorithm (Figure 3a) versus a control algorithm (Figure 3b) that tries to infer the system using randomly generated tests [3]. Each dot corresponds to a positive classification; blank areas correspond to negative classifications. As can be seen in Figure 3b, these 30 hidden systems return very few positive classifications, but the proposed algorithm, after about 100 experiments, tends to approximate the hidden system well enough for evolve queries that extract positive classifications from the target system (Figure 3a). After 300 experiments, on average, the approximation is good enough to allow prediction of sentences that produce positive classifications alternated with negative classifications, implying the algorithm has inferred models that are consistently predictive.

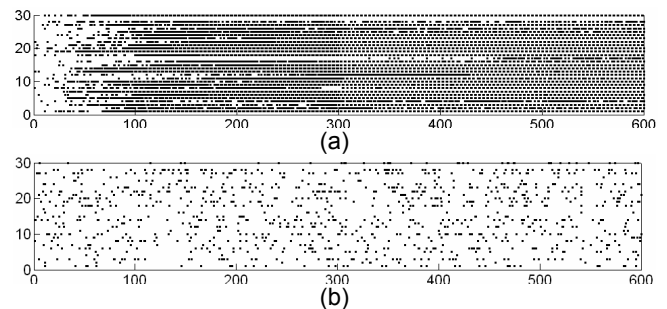


Figure 3. Inference of finite state machines. Resulting classifications for 30 independent runs of the (a) proposed algorithm versus (b) a control algorithm that tries to infer the system uses randomly generated tests.

4.2 Inference of robot morphology

A second study that demonstrates the applicability of the exploration-estimation algorithm to the inference of hidden models are experiments carried out on inference of mechanical robot morphologies. Here, the exploration phase evolves tests (candidate controllers) that allow the robot to perform locomotion; the output from the system is the resulting sensor data. The estimation phase then evolves models of the robot morphology (mechanical and electrical) that progressively approximate the target hidden physical machine. The proposed algorithm is able to infer important, yet hidden, aspects of the morphology with a minimum of hardware trials. This information can be used to make models more predictive, and thus significantly reduce the number of physical tests needed.

A quadrupedal robot was evaluated for a fixed duration in a full three-dimensional dynamical simulator, and robots were selected for walking forwards as far as possible in that time. The exploration phase encoded modifications of the controller as a set of 68 synaptic weights that dictates how sensor values are translated into motor commands, through an artificial neural network. The estimation phase encodes some aspects of the robot's morphology. In one study [8], we used the proposed algorithm to recover 17 unknown parameters of the target morphology regarding mass distribution and sensory lag times. From a sample of 30 independent runs it was found that in all cases, and after only 20 physical tests, the 17 parameters of the machine

were recovered sufficiently to allow for good transfer of behavior from the simulated robot to the target robot. In this application, the transfer of behavior serves as a validation of the inference. In a second study [4], the entire morphology was successfully evolved from scratch (Figure 4b)

4.3 Inference and synthesis for actuator and sensor damage compensation

Just as actions can be selected by their ability to create disagreement among candidate models in order to elucidate uncertainties about the world, they can be selected for their ability to elicit desired behavior in agreement among different models, for if different models agree on the outcome of a specific action, then that action is avoiding the uncertainty expressed by the models. The objective of this experiment was to restore operation to a damaged robot. There is no direct information available to the embedded system about the damage that may have occurred, but the functionality must be restored by redesigning a controller that may be qualitatively different than the original controller.

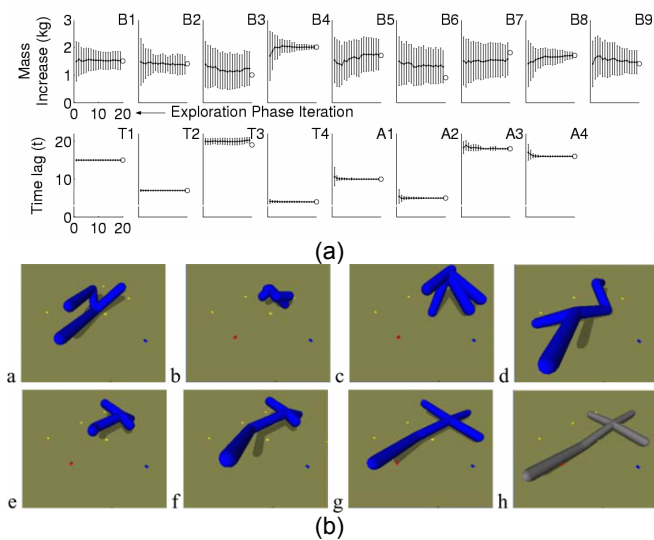


Figure 4. Performance of proposed algorithm to recovery of hidden robot morphology.

Case	Description
1	One motor weakens by 50%.
2	One body part increases in mass by 200%
3	One of the entire legs breaks off.
4	One of the entire legs breaks off, and a sensor fails by 50%.
5	An angle sensor fails by 50%.
6	One of the joints jams by 50%.
7	One of the entire legs breaks off, and one of the joints jams by 50%.
8	One of the entire legs breaks off, one of the joints jams by 50%, and one of the sensors breaks by 50%.
+9	Nothing breaks.
10	The robots stands on a 30 degree horizontal slope.
11	One of the hidden neurons fails by 50%.
12	Two motor neurons output the same value.
13	A body part decreases in mass by 50%.

Table 1. List of damage cases automatically inferred using co-evolution. Cases 10-13 are unanticipated, i.e. are not spannable using damage search space.

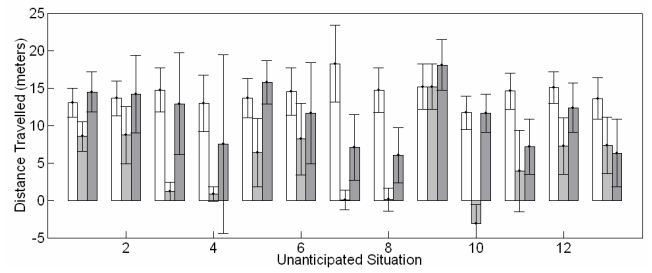


Figure 5. The average performance of the presented algorithm for anticipated and unanticipated situations

In order to recover functionality for a damaged robot, the exploration-estimation algorithm was used to model the damage suffered by a physical robot (which is also simulated in our current work). Three hundred independent runs were executed, in which the 'physical' robot underwent various types of damage: failure or sensor or motors; separation of body parts; changes in mass distribution; actuated joints become jammed; or there is some change in the robot's environment, such as a horizontal tilting of the ground plane. Thirty independent runs for each of 10 different compound failure scenarios were executed.

The progress of a typical run is shown in Figure 6, where function recovery is achieved after a correct diagnosis of the failure (partial sensor failure) is evolved. The average function recovery of the robot for 10 scenarios listed in Table 1, plus three scenarios that cannot be perfectly approximated by the estimation phase, are shown in Figure 5. As can be seen, in almost all cases, function is restored to the crippled robot.

4.4 Inference of general hidden dynamics systems

Inference of the parameters and/or topology of robotic systems and finite state machines can be thought of as special cases of the general problem of inferring a hidden dynamical system based on observation (and control) of its inputs and outputs. Several system-identification methods have been developed for estimating parameters of linear systems [22], but inferring hidden nonlinear systems, and especially their topology, remains a challenge.

The following results demonstrate the use of co-evolution for inference of a symbolic representation of a hidden nonlinear dynamical system. The target system is expressed as a set of general differential equations. The system used here is known as the "Two Eyed Monster" [31], a chaotic system comprising two nonlinear differential equations shown in Figure 7a. For a general dynamical system, we consider the initial state of the system as its input, and the time series describing its behavior as its output. To infer the system we applied the estimation-exploration algorithm evolving candidate symbolic system equations and candidate tests in the form of initial conditions. The symbolic models were represented as parse trees whose nodes are algebraic, trigonometric and exponential operators, and terminals are constants and state variables. This representation is often used in genetic programming to represent symbolic expressions [20].

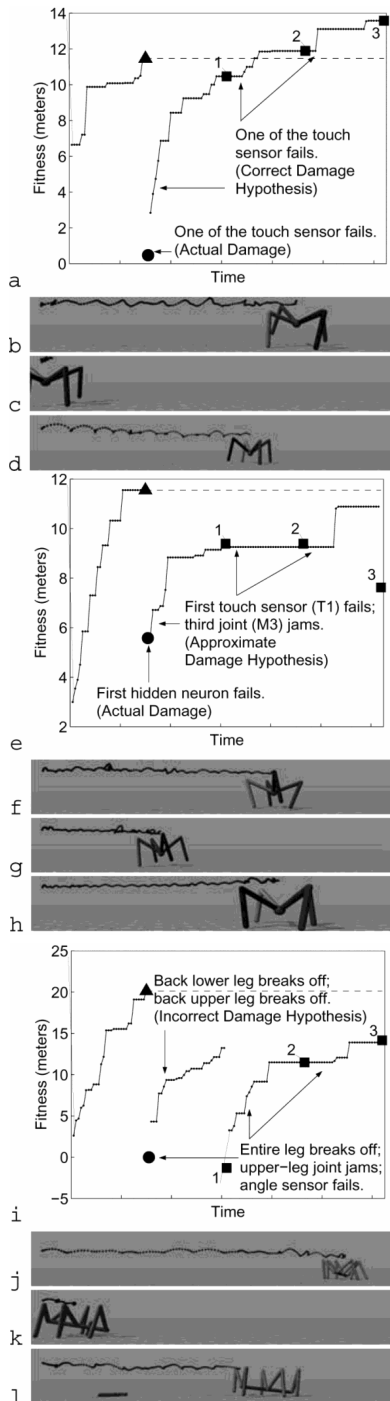


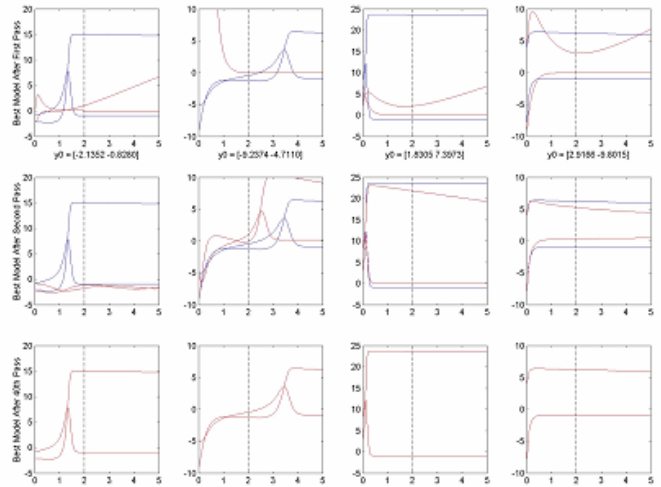
Figure 6. Using inferred models to produce compensatory behavior

The blue curves in the four charts across each row of Figure 7b show the trajectory of the system for 4 different initial conditions. The same four initial conditions are used for the blue curves in every row, but the red curves show the behavior of the best inferred model at various stages. Initially (1st row) the best model is not able to predict (match) the actual performance of the real system (red). In the second row, models are improving in their predictive ability. Finally, in the third row (40th cycle through the exploration-estimation algorithm, after only 40 evaluations of the target system), the models are identical. Figure 7c shows the actual expressions produced. After factoring terms, it is evident that the inferred system is symbolically identical to the target system.

$$x' = y + y^2$$

$$y' = -x + \frac{1}{5}y - xy + \frac{6}{5}y^2$$

(a)



(b)

$$x' = (((y(2)) * (y(2))) + (y(2)))$$

$$= y + y^2$$

$$y' = (((sin((t) - (t))) - (((t) - (y(2))) * (y(2)))) - (y(1))) - (((p(4)) - (y(2)))/(p(2))) + ((y(1)) - (t)) * (y(2))))$$

$$= (((0 - ((t - y) * y) - x) - (((-1 - y)/5) + (x - t) * y))$$

$$= -ty + y^2 - x - ((-\frac{1}{5} - \frac{y}{5} + x - t) * y)$$

$$= -ty + y^2 - x - (-\frac{y}{5} - \frac{y^2}{5} + xy - ty)$$

$$= -x + \frac{1}{5}y - xy + \frac{6}{5}y^2$$

(c)

Figure 7. Symbolic inference of hidden dynamic systems. (a) The target hidden system **(b)** Three rows showing stages of interference: Blue curves show target system for 4 different starting conditions (left to right), red curves show successive approximation. **(c)** The final inferred model after only 40 evaluations is equivalent to the target system.

4.5 Addressing Coevolutionary pathologies

As in a typical co-evolutionary algorithm, the pathologies of red-queen effect and disengagement discussed in section 2.3 occur here too.

The red queen effect is transient in our method because of the anchoring provided by the fixed target system. As the true objective correctness of models improves they are ultimately able to describe test data and therefore their subjective fitness also increases. Conversely, arbitrary overspecialization is removed because it is a source of disagreement among models and is therefore challenged by new tests.

Cycling and transitive dominance effects are eliminated by the fixed target system, because all models need to explain all data so far in addition to any new data.

Disengagement effects may occur when a test may be presented which is too difficult for the model population to explain, and all models get an equally low subjective fitness.

We have recently proposed several mechanisms that successfully prevent disengagement from occurring in the estimation-exploration algorithm.

The “test bank” and evolution roll-back

The first mechanism for addressing coevolutionary disengagement is the “test bank” in which difficult tests are withdrawn from the test suite and saved in a “bank”, and are then only reintroduced later when models become accurate enough to explain them [9]. According to this method, during each cycle the estimation phase continues until there is no more progress in estimation (i.e. model error or fitness stagnates for a number of consecutive generations). If the error is still too high at that point, then the test data is apparently too difficult for the population of models to explain, and disengagement has occurred, possibly resulting in drift. The entire population of models is then “rolled back” to its original state before the new test data was introduced, and an alternative test is generated. If the new test is too difficult, it is banked too; if it is explained by the models, then the banked tests are withdrawn and tried again. The results for identifying the chaotic system in Figure 7 were obtained using this method.

Figure 8 shows the subjective and objective error (inverse of fitness) of a population of models trying to estimate a hidden nonlinear system. In the initial two phases, the subjective error increases but the objective error decreases – this is an example of a transient “red queen” effect. However, the third test produces high subjective error suggesting that it may be too difficult. The test is banked, and the population of model is rolled back to its original state prior to the introduction of the last test. A new test is generated, this time with a subjective error that is much lower. The successive depositing and withdrawing of test data from the bank is evident through the “zigzagging” of the error metric, until the system is able to explain all test data below an error threshold of 0.02.

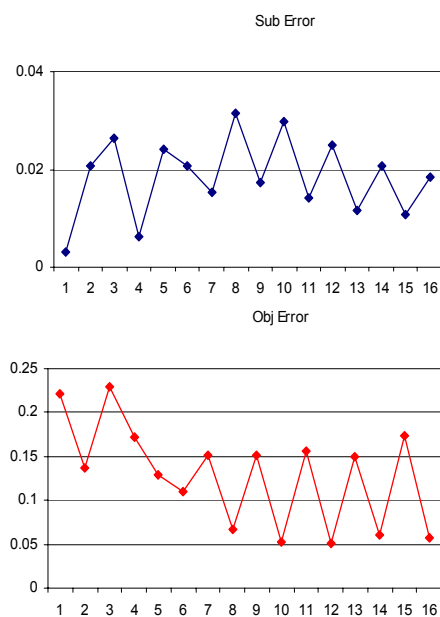


Figure 8. Using ‘test bank’ and evolution rollback to manage co-evolutionary disengagement

Using co-evolutionary variance as a predictor of test difficulty

The second mechanism for addressing coevolutionary disengagement involves searching *explicitly* for lower difficulty tests by looking for tests which create less disagreement among models, and reducing the disagreement until the population re-engages [34]. While test banking and rollback help avoid disengagement, it is not always clear how to generate a new test which will be explicitly easier for the population of models to learn.

Examination of the process reveals, however, that when evolving tests to maximize the disagreement (variance) among predictions of models, they become the most difficult to explain; on the other hand, tests that cause no disagreement are typically easy to explain, and have little instructive value. This relationship suggests that there may be a correlation between the variance that a candidate test produces in the prediction of candidate models, and how “difficult” it will be to learn. This correlation has an information-theoretic justification as the amount of new information a test will yield is proportional to its unexpectedness as indicated by prediction variance.

Figure 9 shows a scatter plot demonstrating a strong correlation between the model variance that a test produces, and the difficulty of explaining it, measured as the best model error using that test. This strong correlation suggests that the variance can be a *predictor* of test difficulty, and can be used to explicitly evolve tests that are appropriate for guiding the estimation phase. This departs from the metaphor of tests as “exams” that serve to differentiate between models, and moves towards the notion of tests being “teachers”, explicitly guiding the progress of models. According to this new approach, if a test is too difficult it is banked, and a new test of half the variance is evolved. If the new test is learned successfully, the previous banked test is withdrawn from the bank; if the bank is empty, a new test of maximum variance is evolved.

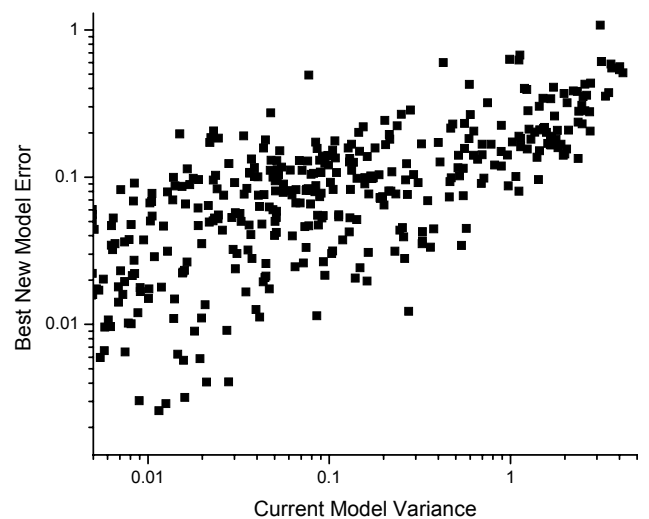


Figure 9. Correlation between variance in model predictions for a candidate test and the difficulty of learning the data produced by the test

5 CONCLUSIONS

We have introduced the estimation-exploration algorithm, a systematic, domain independent method for performing synthesis or analysis using coevolution. Unlike other coevolutionary systems, the estimation exploration algorithm evolves candidate models that are influenced not only by the competing population of tests, but also by the target system. This removes potential pathologies (cycling and the red queen effect) which are often encountered by other coevolutionary algorithms. The third pathology, disengagement, is managed using rollback and disengagement prediction using variance.

The estimation-exploration coevolutionary algorithm allows us to synthesize models and tests for systems about which little is known. This is important in nonlinear black-box system identification tasks for which not even the underlying topology of the hidden system is known. Artificial evolution could be used to build explicit models directly from observed data in such instances, a challenge that has not yet been addressed in the system identification literature. For example the algorithm could be used in a remote robotics application to synthesize a model of a novel environment based only on the robot's sensor data: this is an attractive avenue for future study. In future work we also plan to apply the algorithm to a range of actual physical systems, including physical robots and real biological networks, and further explore implications to design automation.

6 ACKNOWLEDGEMENTS

This work was supported by the U.S. Department of Energy, grant DE-FG02-01ER45902. This research was conducted using the resources of the Cornell Theory Center, which receives funding from Cornell University, New York State, federal agencies, foundations, and corporate partners.

7 REFERENCES

1. Bongard J. C., Lipson H. (2004), "Automated Damage Diagnosis and Recovery for Remote Robotics", IEEE International Conference on Robotics and Automation (ICRA04), pp. 3545-3550
2. Bongard J. C., and Lipson H., (2004) "Automated Robot Function Recovery after Unanticipated Failure or Environmental Change using a Minimum of Hardware Trials", NASA/DoD conference on Evolutionary Hardware 2004, pp. 169-176
3. Bongard J. C., Lipson H. (2004) "Grammatical Inference with Minimal Trials", Journal of Machine Learning Research, in review (revision stage).
4. Bongard J. C., Lipson H. (2004) "Integrated Design, Deployment and Inference for Robot Ecologies", Robosphere 2004, NASA Ames, November 2004.
5. Bongard J. C., Lipson H., (2004) "Automated Robot Function Recovery after Unanticipated Failure or Environmental Change using a Minimum of Hardware Trials", NASA/DoD conference on Evolutionary Hardware 2004, pp. 169-176
6. Bongard J. C., Lipson H., (2004) "Automating Genetic Network Inference Using a Very Low Sampling Estimation-Verification Evolutionary Algorithm", Genetic and Evolutionary Computation Conference, (GECCO '04), pp. 333-345.
7. Bongard J. C., Lipson H., (2004) "Nonlinear System Identification using Co-Evolution of Models and Tests", IEEE Transactions on Evolutionary Computation, in review (revisions stage)
8. Bongard J. C., Lipson H., (2004) "Once More Unto the Breach: Automated Tuning of Robot Simulation using an Inverse Evolutionary Algorithm", Ninth Int. Conference on Artificial Life (ALIFE IX), pp. 57-62.
9. Bongard J., Lipson H.. (2005) 'Managed challenge' alleviates disengagement in co-evolutionary system identification. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05), submitted, 2005
10. Bongard, J. C. (2002) Evolving Modular Genetic Regulatory Networks, in Proceedings of the IEEE 2002 Congress on Evolutionary Computation (CEC2002), IEEE Press, pp. 1872-1877.
11. Bucci A., Pollack J.B., De Jong E.D.. (2004) Automated extraction of problem structure. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004), pages 501–512. Springer-Verlag
12. Cicchello O., Kremer S. C., (2003) Inducing Grammars from Sparse Data Sets: A Survey of Algorithms and Results, Journal of Machine Learning Research 4, pp. 603-632
13. Cliff D., Miller G.F., (1995) Tracking The Red Queen: Measurements of adaptive progress in co-evolutionary simulations, European Conference on Artificial Life 95
14. De Jong E.D. Pollack J.B. (2004). Ideal evaluation from coevolution. Evolutionary Computation, 12(2):159–192
15. Ficici S.G., Pollack. J.B (2001) Pareto optimality in coevolutionary learning. In Advances in Artificial Life: 6th European Conference (ECAL 2001), pages 316–327. Springer Verlag
16. Goldberg D.E., (2002) *The Design of Innovation*, Springer
17. Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C. et al. (Eds.), Artificial Life II. Addison Wesley
18. Juillé, H. (1999). Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm. Doctoral Dissertation, Brandeis University, Department of Computer Science, May 1999
19. Juille, H., Pollack, J. B. (1998). A stochastic search approach to grammar induction. In: Procs. Of the Fourth International Colloquium on Grammar Inference. pp. 126—137.

20. Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press
21. Lipson H., Bongard J. C., (2004) "An Exploration-estimation algorithm for synthesis and analysis of engineering systems using minimal physical testing", ASME Design Automation Conference (DAC04)
22. Ljung, L., (1999) *System Identification: Theory for the User*, Prentice-Hall Inc., Englewood Cliffs, NJ
23. Mahfoud S.W., Niching (1996) methods for genetic algorithms, Doctoral Dissertation, University of Illinois at Urbana-Champaign
24. Olsson B.. (2001) Co-evolutionary search in asymmetric spaces. *Information Sciences*, 133:103–125
25. Papadimitriou C. H, Steiglitz K, (1982) *Combinatorial Optimization : Algorithms and Complexity*, Dover
26. Paracer S., Ahmadjian V. (2000) *Symbiosis: An Introduction to Biological Associations*, Oxford University Press; 2nd edition
27. Parekh, R., Honavar, V. (2001) DFA learning from simple examples. *Machine Learning* 44: 9-35.
28. Potter M.A., De Jong K.A.. (2000) Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29
29. Rosin C. D., Belew. R. K. (1997) New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.
30. Stanley K.O. and Miikkulainen R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100
31. Strogatz S. H., (2001) *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*, Perseus Books Group
32. Watson R.A. and Pollack J.B. (2001). 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.
33. Watson, R.A., Pollack, J.B. (2003). A Computational Model of Symbiotic Composition in Evolutionary Transitions, *Biosystems*. 69 (2-3), pp.187-209
34. Zykov V., Bongard J., Lipson H. (2005). Co-evolutionary variance guides physical experimentation in evolutionary system identification. In *The 2005 NASA/DoD Conference on Evolvable Hardware*, submitted, 2005.