Journal of Machine Learning Research 6 (2005) 1651–1678 Submitted 8/04; Revised 6/05; Published 10/05

Active Coevolutionary Learning of Deterministic Finite Automata

Josh Bongard JOSH.BONGARD@CORNELL.EDU
Hod Lipson HOD.LIPSON@CORNELL.EDU
Computational Synthesis Laboratory
Sibley School of Mechanical and Aerospace Engineering
Ithaca, NY 14853, USA

Editor: Stefan Wrobel

Abstract

This paper describes an active learning approach to the problem of grammatical inference, specifically the inference of deterministic finite automata (DFAs). We refer to the algorithm as the estimation-exploration algorithm (EEA). This approach differs from previous passive and active learning approaches to grammatical inference in that training data is actively proposed by the algorithm, rather than passively receiving training data from some external teacher. Here we show that this algorithm outperforms one version of the most powerful set of algorithms for grammatical inference, evidence driven state merging (EDSM), on randomly-generated DFAs. The performance increase is due to the fact that the EDSM algorithm only works well for DFAs with specific balances (percentage of positive labelings), while the EEA is more consistent over a wider range of balances. Based on this finding we propose a more general method for generating DFAs to be used in the development of future grammatical inference algorithms.

Keywords: grammatical inference, evolutionary computation, deterministic finite automata, active learning, system identification

1. Introduction

Grammatical inference is a popular machine learning domain (refer to Cicchello and Kremer, 2003, for an overview): it has wide applicability in both computational linguistics and related fields, as well as giving rise to a host of benchmark problems (Tomita, 1982; Lang et al., 1998) and competitions. Grammatical inference is a special case of the larger problem domain of inductive learning (Bergadano and Gunetti, 1995), which aims to construct models of some underlying system based on sets of positive and negative classifications. In one class of grammatical inference methods, the system is considered to be some kind of language or classifier, and models are represented as deterministic finite automata (DFA). Both the target system and models take strings of symbols as input (sentences), and produce binary classification as output (labellings), indicating whether that sentence belongs to the language or not. The problem of grammatical inference can also be considered a special instance of the problem of system identification (Ljung, 1999), in which some target system is inferred based solely on input/output data.

Grammatical inference methods that employ DFAs as models can be divided into two broad classes: passive and active learning methods. In passive methods, a set of training data is supplied to the algorithm for model construction. In active learning approaches, the algorithm has some

influence over which training data is labeled by the target DFA for model construction. Active learning approaches are typically iterative, in which membership queries are proposed periodically, often in response to some deficiency in the currently constructed models. In these iterative active approaches the amount of training data available for inference grows over time, unlike passive approaches, in which a fixed set of training data is used for model construction.

Passive methods usually make some assumption about the training data: a set of labeled training data is either generated by some auxiliary method randomly, or according to some predefined distribution. For example Pitt (1989), Porat and Feldman (1991), Dupont (1996) and Lang et al. (1998) assume a randomly-selected set of sample data; Luke et al. (1999) and Lucas and Reynolds (2005) assume equal amounts of positive and negative training data when inferring the Tomita languages (Tomita, 1982) by using the same training sets as previous researchers; Pao and Carr (1978) and Parekh and Honavar (1996) assume a structurally complete set; Oncina and Garciá (1992) assume a characteristic sample; and Angluin (1981) assumes a live complete set. Once the sample data has been generated and labeled, inference is then conducted.

With the exception of randomly-generated training data, it is assumed that the training data is collected using some knowledge of the target system to be inferred. For example one necessary criterion for a structurally complete set of training data is that it covers every state transition of a DFA[1] (Pao and Carr, 1978; Parekh and Honavar, 1993; Dupont et al., 1994). This requires that the algorithm which generates the training data knows something about the structure of the DFA, namely its state transitions. This is advantageous as then it is possible to make performance guarantees regarding an inference algorithm working on that training data. However, for real-world usage of grammatical inference algorithms, it is unreasonable to assume that the internal structure of the DFA is known: indeed, this is exactly what is being inferred. In this work we present an active learning algorithm that makes few assumptions about the structure of the target DFA, and in fact outperforms one of the best heuristic methods for grammatical inference, which implicitly assumes that the DFAs are balanced (i.e. produce a more or less equal number of positive and negative labelings).

The current most powerful passive approach to grammatical inference using DFAs as models are the evidence driven state merging (EDSM) methods (see Cicchello and Kremer, 2003, for an overview), a heuristic approach that iteratively compresses an initially large DFA down to a smaller one, while preserving perfect classification before and after each compression. In this paper we compare our algorithm's performance against an EDSM variant implemented by Lucas and Reynolds (2005). Evolutionary approaches to grammatical inference also exist, in which a stochastic search method seeks the most accurate DFA model through mutation and recombination of previous models: in this work we will also compare our own method, which employs evolutionary computation for search, against the evolutionary method proposed by Lucas and Reynolds (2005). However, like the other passive methods, both heuristic and evolutionary approaches so far assume that some external agent generates either a random or balanced training set[2] before inference begins.

In the active learning approach to regular language inference pioneered by Angluin (1987) (see also Berg et al., 2003, and Angluin, 2004), the algorithm iteratively requests membership queries for training data it has generated on its own. Despite this active approach to training data generation, these algorithms also require an external agent—an oracle—that can answer equivalence queries: the oracle indicates whether the current model is equivalent to the target DFA and, if it is not, returns

1. See the definition of states and state transitions in Section 2.1 below.
2. A training set containing an equal number of positive and negative samples.

new training data that belongs to the target language but does not belong to the language encoded by the candidate model. Once again, this assumes that the oracle knows something about the structure of the target DFA. The algorithm presented here assumes that an oracle can answer membership queries, but not equivalence queries. In practical applications, such an oracle is the target system itself: the target system will return a classification for a proposed item, but cannot indicate whether a proposed model is equivalent to itself or not. The target system can indicate the goodness of a model if a large amount of sample data is classified by both itself and the proposed model and the resulting classifications are compared, but for target systems in which classifications are costly, slow or dangerous, this is not feasible.

Other active learning approaches to language inference also exist, but they all assume completely passive reception of training data: Sempere and Garcia (1993) only require that samples be presented in lexicographic order, and the RPNI (Oncina and Garciá, 1992, and Lang et al., 1992) and RPNI2 Dupont (1996) algorithms assume random training data is supplied by an external agent, with the stipulation that positive and negative sample data must be made available.

The method presented in this paper does not assume any passive reception of training data from an external agent: rather, the algorithm attempts to evolve sentences that, when passed to the target system, should indirectly extract information about previously hidden components of the target system. For example, sentences should be sent to a target system that, during labelling, cause transitions to states that have never or rarely been visited during previous labellings. This is particularly useful in cases when passively-generated training data will cause some states of the target DFA to be visited much more often than others. In system identification, such systems are said to have low observability; it is more difficult to observe some components of the system than others using input data generated without recourse to a partial model of the system. For this and other reasons, it is not surprising that active learning approaches outperform passive methods: active methods have more control over the collection of training data. However the point of this paper is to demonstrate one reason why active methods outperform passive methods: namely, that they perform well on both balanced and imbalanced DFAs. More specifically, it is shown that one of the leading passive methods, the EDSM method, does poorly because it only performs well on balanced DFAs using balanced training data.

Large and unbalanced DFAs are one kind of automata that have low observability: these DFAs contain a large number of states, but tend to produce one labelling much more often than the other labelling, for any given sentence. For example one particular language (Tomita language 1, see Tomita, 1982) only produces a positive classification for a given binary string 2.4% of the time. In such cases, generating random training data is not recommended, because few or no sentences that elucidate the pathways to accepting states will be collected. Also, generating balanced training data is also not recommended, for two reasons. First, there will be a surfeit of training data elucidating paths to accepting states, and most likely not enough training data to elucidate the many other paths to non-accepting states, leading to the generation of a model that may have high training data accuracy but low test set accuracy. Secondly, generating balanced training data requires many labellings by the target system until a sufficient number of the minority labellings are collected. For example in order to obtain training data with 100 positively labelled data and 100 negatively labelled data for Tomita language 1, at least $\lceil \frac{100}{0.024} \rceil = 4167$ labellings of randomly generated sentences would have to be performed. This is not desirable for the real-world inference of languages or classifiers for which it is costly, dangerous or slow to perform a target labelling: the two performance

metrics for grammatical inference are model accuracy, and a minimum of sentence labellings by the target system.

Here we show that our algorithm outperforms competing methods that assume randomly-generated training data. For the case of imbalanced DFAs, we attribute this performance improvement to the discovery of sufficient minority class training data to produce accurate models: randomly-generated training data contains too little minority class training data. We support this claim by showing that the proposed algorithm performs well over a range of DFAs with differing balances (percentage of positive labellings), but that the EDSM method implemented here only performs well on DFAs within a narrow range of balances.

The fact that our algorithm also outperforms competing algorithms on balanced DFAs suggests that those DFAs contain state transition pathways that are rarely traversed by randomly-generated training data, but are better traversed by our proposed algorithm. However as of yet we have no supporting evidence for this stronger claim.

In the next section we briefly describe grammatical inference, as well as describing our method for the inference of target DFAs using active training data generation. We also document an evolutionary and a heuristics-based method for performing grammatical inference using pre-selected training data. In Section 3 we compare results from our algorithm against these algorithms for both randomly-generated DFAs, and randomly-generated DFAs that have differing balances. In the final section we provide some discussion and concluding remarks.

2. Methods

In this section we introduce grammatical inference, and outline three methods for approaching the problem: evidence-driven state merging, evolutionary approaches, and the estimation-exploration algorithm.

2.1 Grammatical Inference

A deterministic finite automata, or DFA, is a type of finite state automata that can be represented using the five-tuple (n, Σ, T, s, F) where n is the number of states, Σ is the alphabet of the encoded language, T is a transition function, s is the start state, and F is a set of final, or accepting states. Then, given some sentence made up of a string of symbols taken from the alphabet Σ, and beginning at the start state s, the first symbol is extracted from the sentence, and based on that symbol the sentence transitions to a new state as indicated by T. A deterministic finite automata follows the transition dictated by the current sentence symbol, the current state and the state transition function T with a probability of 1; probabilistic finite automata (which have not yet been investigated using our method) include probability distributions that denote the probabilities of transitioning to new states given the current sentence symbol, the current state and the state transition function.

After a state transition the next symbol is then extracted from the sentence, and based on T the sentence transitions to a new state. This process is continued until all symbols in the sentence have been exhausted. If the last state visited is a member of F, then the sentence receives a positive classification (the sentence belongs to the language); otherwise, a negative classification is assigned (the sentence does not belong to the language).

The quality of a grammatical inference algorithm is viewed as one that can produce some T' and F' (together referred to as a candidate DFA) that matches the labels of a pool of sentences that have already been labelled by the target DFA (the training set accuracy). The candidate DFA

should then produce a high classification accuracy when supplied with a different set of unlabelled sentences (test set accuracy). More specifically, quality can be measured in five ways: generation of an accurate model using a small set of training data; probability of learning the target language using little training data; continued performance for target DFAs with increasingly more states; consistent performance across DFAs with differing balances; and generation of an accurate DFA in the face of training set noise.

2.2 Evidence-Driven State Merging Algorithm

A family of algorithms collectively known as evidence-driven state merging algorithms (EDSMs) (Trakhtenbrot and Barzdin, 1973; Lang et al., 1998; Cicchello and Kremer, 2003) have been proposed that can infer some target DFA in which the number of states is unknown. EDSM algorithms operate by first generating an augmented prefix tree acceptor (APTA) which by definition perfectly classifies all sentences in the training set. Subsequent steps then involve merging states such that the DFA still maintains perfect training set accuracy. It has been shown that state merging tends to increase the test set accuracy of the reduced DFA. However in the face of incomplete training data, it is possible that an incorrect merge may occur: a merge that does not affect training set accuracy but does decrease test set accuracy. All of the work on EDSM algorithms is concerned with how merges should be performed in order to minimize test set accuracy degradation.

The EDSM method employed in this paper is adopted from (Lucas and Reynolds, 2005), which in turn modifies the EDSM method proposed by Price (Lang et al., 1998). The algorithm works as follows. Each pair of states in the APTA (or partially folded APTA) are considered for merging. The score of each merge is calculated by overlapping the roots and subtrees of the selected state pair, and summing the number of overlapped states that are either both accepting or rejecting states. If an accepting and rejecting state are found to overlap, that merge is disqualified. The state pair with the highest score is then selected for merging. In the case of a tie score, the state pair that appears first in the sequence of upper triangular matrix raster scan order $[(0,1),(0,2),\ldots,(1,2),(1,3),\ldots]$ is selected. The merge is then performed, and the previous steps are repeated until no further merges can be performed (*i.e.* all candidate merges are disqualified).

2.3 Evolutionary Approaches to Grammatical Inference

Evolutionary approaches to grammatical inference have also been proposed (Brave, 1996; Luke et al., 1999; Lucas and Reynolds, 2005). Generally, an evolutionary algorithm comprises a population of candidate models of the target DFA that compete against each other, and the fitness of a particular model is given by the percentage of training data that it can correctly classify. The model with the highest fitness at the termination of the run is then evaluated against the unlabelled test data. In this paper we compare our own evolutionary algorithm against the evolutionary method proposed by Lucas and Reynolds (2005). This approach is described below.

2.3.1 EVOLVING DFAS WITH A FIXED NUMBER OF STATES

Lucas and Reynolds (2005) proposed an evolutionary approach to grammatical inference in which the number of states in candidate models is fixed at $5n/4$, where n is believed to be the number of states in the target DFA. On target DFAs with $n \leq 16$ and a range of training set densities, this methodology outperforms the EDSM method outlined above (see Section 3).

1. **Characterization of the target system**
   - Define a representation, variation operators and similarity metric for the space of systems
   - Define a representation and variation operators for the space of inputs (tests)
   - Define a representation and similarity metric for the space of outputs
2. **Initialization**
   - Create an initial population of candidate models (random, blank, or seeded with prior information)
   - Create an initial population of candidate tests (random, or seeded with prior information)
3. **Estimation Phase**
   - Evolve candidate models; encourage diversity
   - Fitness of a model is its ability to explain all input-output data in training set
4. **Exploration Phase**
   - Evolve candidate tests (input sets)
   - Fitness of a test is the disagreement it causes among good candidate models
   - Carry out best test on target system, add input/output data to training set
5. **Termination**
   - Iterate estimation-exploration (steps 3-4) until the population of models converges on a sufficiently accurate solution, or the target system exhibits some desired behavior.
   - If no model is found, the search space may be inappropriate, or the target system may be inconsistent
   - If no good test is found, then either:
     - all good candidate models are perfect;
     - the search method for finding good tests is failing; or
     - the target system may be partially unobservable
6. **Validation**
   - Validate best model(s) using unseen inputs
   - If validation fails, add new data to training set and resume estimation phase

Table 1: Estimation-Exploration Algorithm Overview

In this method a transition function T' is encoded as a $\Sigma \times \frac{5n}{4}$ matrix, and each element in T', t'_{ij}, lies in the range $[0, \frac{5n}{4} - 1]$. Each column of T' corresponds to a particular state: the first column is regarded as the start state. During parsing, state transition is computed as follows. Transition to the state indicated by t'_{ij}, where i is the current state, and the current symbol from the input sentence corresponds to the jth letter in alphabet Σ.

Lucas and Reynolds (2005) realized that it is not necessary to evolve F' in addition to T', but rather that it can be constructed indirectly from T' and the training data. For each state in T', compute the ratio of positive and negative training sentences that terminate at that state: if more positive sentences terminate there, then consider that state an accepting state; otherwise, consider it a rejecting state.

Rather than employing a generational genetic algorithm, this method employs a multi-start hill climber. A random instance of T' is selected, and then a mutation is introduced: if the mutation causes a decrease in training set accuracy, revert to the original T'. Otherwise, keep the mutation, and perform another mutation. If 10,000 mutations have been attempted with no improvement, store the current T' and create a new random T'. Continue this process until a T' perfectly labels the training data, or until a total of 1,000,000 mutations have been attempted. Return the T' with the highest training set accuracy.

2.4 The Estimation-Exploration Algorithm

Both the heuristic and evolutionary method described above assume passive inference: a set of labelled data is presented to the algorithm, and the algorithm produces a candidate model of the target DFA. We have developed an active learning methodology for inferring DFAs (as well as other nonlinear target systems) which we refer to as the estimation-exploration algorithm (EEA). The algorithm is composed of two phases, as are most active learning systems (refer to Baram et al., 2004, for an overview of active learning): the estimation phase uses i instances of training data obtained from the target system to construct a set of candidate models; the exploration phase generates a new sentence (an instance of training data) that causes maximal disagreement among the candidate models. This new sentence is then supplied to the target system, and the estimation phase then begins again with $i+1$ training data points. The estimation phase in our algorithm corresponds to the learning algorithm $\mathcal{A}$ as described by Baram et al. (2004), and the exploration phase corresponds to the querying function Q. The utility of an active learning system corresponds to how well $\mathcal{A}$ and Q perform together, compared to a control method where $\mathcal{A}$ operates alone on randomly-generated unlabelled data (refer to Baram et al. (2004) for an overview of active learning).

The estimation-exploration algorithm is essentially a $co-evolutionary$ process comprising two populations. One population is of candidate models of the target system, where a model's fitness is determined by its ability to correctly explain observed data from the target system. The other population is of candidate unlabelled sentences, each of whose fitness is determined by its ability to cause disagreement among model classifications (thereby elucidating model uncertainties), or by exploiting agreement among models to achieve some desired output (thereby capitalizing on model certainties). The query by committee algorithm (Seung et al., 1992) first proposed that a good test is one that causes maximal agreement among a set of different candidate learners: however, the method by which differing yet accurate learners and disagreement-causing tests are generated was not given. In the estimation-exploration algorithm, evolutionary algorithms are used both to synthesize accurate yet differing models, as well as useful tests. If successful, the two populations challenge each other and drive an 'arms-race' towards inference of the model or towards eliciting some desired output from it. In previous papers (Bongard and Lipson, 2004b,a) we outlined a methodology for applying the estimation-exploration algorithm to other kinds of nonlinear target systems. The general methodology is given in Table 2.3.1, and the specific application to grammatical inference is given below.

2.4.1 CHARACTERIZATION OF THE TARGET SYSTEM

Like Lucas and Reynolds (2005), we choose to represent the target DFA and candidate models as $2 \times n$ integer matrices. For target DFAs with alphabets containing more than two elements, a larger matrix or a different encoding would be required. For the case of the target DFA, n is known. For

the work presented here, however, we assume that the learning algorithm does not know the number of states in the target system, but has some idea as to the upper bound. For the random target DFAs presented in Section 3.1, we compare our results against those of Lucas and Reynolds (2005), in which the number of states in a candidate model was fixed to be $\frac{5n}{4}$: we set the maximum number of states in a candidate model to be $2n$ in order to make less assumptions about the size of the target DFA. The second difference between our method and that of Lucas and Reynolds (2005) is that we exert selection pressure favoring smaller DFAs, in the hope of discovering more general models, as will be explained in the subsection documenting the estimation phase below.

It is always assumed that the first state is the start state. In addition, the target DFA contains an additional binary vector of length n that indicates whether state i is an accepting or rejecting state (F). For each candidate DFA model T' we compute F' using the method described in Section 2.3.1, as originally proposed by Lucas and Reynolds (2005). Due to the difficulties of devising a similarity metric between the target DFA and a given candidate model, we have chosen to denote similarity between a model and target DFA as the test set accuracy of the candidate model. If it were possible to define a similarity metric between the target and a model DFA, it would be possible to quantitatively determine how well an inference algorithm was doing by periodically measuring the similarity of candidate models against a target DFA. This would serve as a validation phase before using the algorithm in a practical application, where it is assumed there is no measure of target-model similarity, except for a model's ability to consistently match the classifications produced by the target.

In the exploration phase, for the grammatical inference problem a training item is considered to be an unlabelled binary sentence s'. Each sentence is represented as a binary vector of length $s_{\max}$, where $s_{\max}$ is the maximum sentence length to be found in the training or test set. An additional integer variable l is selected from $[0, s_{\max}]$ with a uniform distribution, and indicates how long the encoded sentence is. If $l < s_{\max}$, then the trailing digits $[l, l+1, \ldots, s_{\max}]$ are ignored during the labelling of the sentence.

2.4.2 INITIALIZATION

The algorithm begins inference by generating an unlabelled sentence at random, which is then labelled by the target DFA. The training set, consisting of a single labelled sentence, is then provided to the candidate models in the estimation phase.

During the first pass through the estimation phase, a random population of candidate models is generated. In order to generate a pool of competing candidate models, the population of models in the estimation phase is partitioned into two equally-sized, reproductively isolated sub-populations: no candidate model can place offspring into the other sub-population. When the estimation terminates, the two most fit candidate models from each sub-population are provided to the exploration phase. This partition has no additional computational costs, as the two populations of models take the same time to evaluate as a single population with twice as many models.

During subsequent passes through the estimation phase, the two best models from the previous pass are introduced into their respective sub-populations. The remaining slots are filled with randomly-generated candidate models.

At the beginning of each pass through the exploration phase, the population is seeded with random binary sentences.

2.4.3 ESTIMATION PHASE

It is important to maintain a diverse set of candidate models in the estimation phase, so that disagreement among models can be measured effectively at the exploration phase. Many diversity maintenance techniques exist, but here we take the simplest approach based on evolving in two, separate niches. Starting with an initial population of p candidate models (with $p/2$ models in each of the two sub-populations), the population is evaluated, fit models are selected, copied and mutated, and less fit models are deleted. No recombination operators are employed in the current implementation. The pass continues for a fixed number of generations (g).

The fitness of a candidate model is given by

$$f_{T'} = 1 - \frac{\sum_{j=1}^{i} |t_j - m_j|}{i}, \quad (1)$$

where t_j is the labelling provided for the jth sentence by the target DFA, and m_j is the labelling provided for the same sentence by the model DFA. Then a candidate model that obtains $f_{T'} = 1$ perfectly labels all of the i training sentences seen so far, and models with lower values of $f_{T'}$ have lower accuracies. Within each sub-population, genomes are then sorted in order of decreasing fitness, such that the model with the highest fitness is at the top of the list of models within the sub-population, and the least fit model is at the bottom of the list. If two or more models have the same fitness, then those models are sorted among themselves based on the number of internal states that were visited during processing of all i training sentences seen so far, such that the topmost model in the subset used the least number of states, and the bottommost model used the most.

Once all of the models in both sub-populations have been evaluated, a pair of candidate models from within the same sub-population are selected. Each genome has an equal probability of being selected. The lower model in the sorted list is overwritten by a copy of the model higher up in the list. This ensures that models with higher fitness produce more offspring than models with lower fitness, and smaller models produce more offspring that models of larger size and equal fitness: selection pressure favors more accurate and more compact models. If both models have the same fitness and the same size, then each model in the pair has an equal probability of replacing, or being replaced by the other model. Also, this selection method ensures that the model with the best fitness and least size in each sub-population is never overwritten.

When a model is copied, it undergoes mutation. Mutation involves the selection of a random value t'_{ij}, and the replacement of the value found there by a new integer chosen from $[0, 2n-1]$ with a uniform distribution.

A total of $\frac{3p}{8}$ pairs are selected for replacement and mutation in each sub-population at the end of each generation. Note that a genome may be selected more than once during the same generation, and that it may produce more offspring, or be overwritten by a more fit or smaller model. Also note that a mutated offspring may be selected for copying and further mutation during the same generation.

2.4.4 EXPLORATION PHASE

The exploration phase maintains a population of the same size as that of the estimation phase (p), and evolves candidate sentences for the same number of generations (g). At the end of each generation, $\frac{3p}{4}$ pairs of sentences are selected, copied and mutated as described in the previous section: the sentence with higher fitness is copied over the sentence with lower fitness, and the copied sentence is then mutated.

The fitness for a given sentence is set to the amount of disagreement that that sentence causes when labelled by a pool of k candidate models:

$$f_{s'} = 1 - 2|0.5 - \frac{\sum_{j=1}^{k} c_j}{k}|, \quad (2)$$

where c_j is the classification of the candidate sentence by model j. Sentences that do not differentiate between the candidate models—all models produce the same classification—obtain $f_{s'} = 0$ (poorest quality); sentences that produce the maximum classification variance obtain $f_{s'} = 1$ (best quality). This fitness function relies on the fact that the most agreement is equivalent to half of the models returning a negative classification, and the other half returning a positive classification. For target DFAs that do not produce binary classifications, this function would have to be generalized.

When a sentence is evolved that induces high classification variance, and that sentence is classified by the target DFA, then the resulting classification will usually lend support to $k/2$ candidate models during the next pass through the estimation phase, and provide evidence against the remaining half. It is important to note that for the experiments reported here, $f_{s'}$ can only assume two values: 0 or 1, hence there is no gradient within the search space. This is the simplest implementation of the algorithm. We expect that increasing the number of sub-populations or using other diversity maintenance techniques such as deterministic crowding (Mahfoud, 1995) would improve these results by inducing a gradient in the search space. Future work is planned to assess the performance benefit of population diversity.

Mutation is executed slightly differently from the estimation phase: with an equal probability, either the sentence or the length parameter l are selected for mutation. If the sentence is selected, a random bit is chosen and flipped; if l is chosen, l is reset to a random value in $[0, s_{\max}]$. In this way the algorithm can modify both the content and length of a candidate string.

2.4.5 TERMINATION

A typical run would terminate when the population of models converges. In the experiments reported here, however, the number of iterations was set to use exactly the same number of sentence labellings as the benchmark algorithms we compare it to require. The algorithm iterates t times, where t is the number of sentences in the training set used by the competing algorithm. This results in the labelling of t sentences by the target DFA, t passes through the estimation phase, and $t-1$ passes through the exploration phase (the first sentence proposed for labelling is a random sentence). After the tth pass through the estimation phase, the most fit model from the first sub-population is output for validation purposes.

2.4.6 VALIDATION

Validation involves computing the accuracy of the best candidate model on a previously unseen set of test sentences.

3. Results

The estimation-exploration algorithm was compared against two sets of target DFAs: DFAs generated randomly in accordance with the method described in (Lang et al., 1998) for generating DFAs with differing sizes; and DFAs generated using a more generalized method that creates DFAs of differing sizes and balances.

3.1 Random DFAs

Sets of target DFAs of increasing size were generated for comparison between the EDSM method described in Section 2.2 (indicated as 'EDSM' in Figure 2), the evolutionary method proposed by Lucas and Reynolds (2005) (indicated as 'Lucas' in Figure 2), the estimation-exploration algorithm with the exploration phase disabled (random sentences are proposed to the target DFA and indicated as 'Passive EEA' in ensuing figures), and the estimation-exploration algorithm (indicated as 'Active EEA' in ensuing figures). Although the passive variant of the EEA proposes sentences to the target DFA for labeling, it is considered passive because it does not actively construct training data; rather, it outputs random training data. This stresses the importance of actively seeking informative sentences for target labeling.

As prescribed by the generative method introduced by Lang et al. (1998), the target DFAs were generated by creating random digraphs with $5n/4$, where n is the desired number of active states: an active state is one that is visited by at least one test or training sentence during labelling. Graphs are continually generated until one is produced which has a depth of exactly $2\log_2 n - 2$, where the depth of a DFA is determined to be the maximum over all states of the length of the shortest string which leads to that state:

$$d = \max_{(x|\frac{5n}{4})} \min_{(y|\text{strings leading to state } x)} \text{length}(y). \tag{3}$$

Once a target DFA is generated, each state is labelled as either accepting or rejecting with equal probability.

The total number of binary strings available for labelling is given as

$$S_{\text{total}} = \sum_{i=0}^{\lfloor (2\log_2 n)+3 \rfloor} 2^i, \tag{4}$$

in accordance with the Abbadingo method (Lang et al., 1998) for generating random DFAs, where $\lfloor (2\log_2 n)+3 \rfloor$ is the maximum possible string length. This approach ensures that all binary strings from the null string to length $\lfloor (2\log_2 n)+3 \rfloor$ can be found in either the training or test set.

Strings selected for membership in the training set for the passive methods outlined in Sections 2.2 and 2.3.1 were selected at random (with a uniform distribution) from among this set of possible strings. Given a desired training set density d, the number of strings chosen for the training set can then be computed as $t = \lfloor dS_{\text{total}} \rfloor$.

In order to fairly compare our active learning method against passive methods, we have elected to equalize the number of labellings performed by the target DFA (*i.e.* the number of training sentences that are labelled), and the number of labellings performed by candidate models during inference. Because EDSM methods do not maintain a population of candidate models but rather iteratively compress a single one, in order to compare our method against the EDSM method outlined here we equalize the amount of labelled data that both algorithms have access to.

In Lucas and Reynolds (2005), the total number of candidate model labellings m is equal to the number of training sentences times the number of mutations considered during hill climbing: $m = \lfloor dS_{\text{total}} \rfloor \times 10^6 = t \times 10^6$, where d is training set density and t is the number of training sentences. In the estimation-exploration algorithm a total of

$$pgk(t-1) + pg\sum_{i=1}^{t} i$$

labellings are performed, where p is the population size and g is the number of generations for both phases, and thus pg indicates the total number of either sentences or models that are evaluated during a single pass through either phase. k indicates the number of candidate models output by the estimation phase,[3] so $pgk(t-1)$ indicates how many labellings are performed during the $t-1$ passes through the exploration phase. The second term indicates how many labellings are performed during the t passes through the estimation phase: during the first pass there is only one labelling per candidate model; during the second pass there are two labellings per model; and so on.

We can ensure that our method performs the same or fewer model labellings as Lucas' method by arbitrarily setting $p = g$, and solving for p as follows:

$$pgk(t-1) + pg\sum_{i=1}^{t} i = m \quad (5)$$

$$p^2(2(t-1) + \sum_{i=1}^{t} i) = t \times 10^6 \quad (6)$$

$$p, g = \lfloor \sqrt{\frac{t \times 10^6}{(2(t-1) + \sum_{i=1}^{t} i)}} \rfloor \quad (7)$$

Note that $k = 2$ here because we partition the estimation phase populations into two sub-populations.

3.1.1 THE EFFECT OF COMPRESSION PRESSURE ON INFERENCE

One advantage of the EEA over the EDSM methods is that it allows for both compression and expansion of models: EDSM methods only allow for compression at each step, and do not allow re-expansion. The advantage of this is illustrated in Figure 1, which reports the application of the EEA to a randomly-generated DFA with $n = 8$ states. Two other variants were also applied to the same DFA. The passive variant disables the exploration phase of the algorithm, so that at each cycle through the algorithm, a random sentence is output to the target DFA for labelling. The third variant is identical to the active variant, except for two modifications. First, the fitness function that favors smaller DFAs is disabled: when two candidate models are selected and both achieve the same training set accuracy, the first model is copied over the second model with a probability of 0.5, regardless of whether the second model is smaller than the first. Second, the maximum number of states that any candidate model can encode in this variant was increased from $2n = 16$ to 80. At the end of each pass through the estimation phase for each variant, the test set accuracy of the best candidate model output by the first sub-population is computed. Only the first 100 iterations through each variant are shown.

As can be seen in Figure 1a, the active variant outputs a model consistent with all training and test data after the 93rd pass through the estimation phase. The other two variants never produce a perfectly consistent model. Figure 1b indicates that the size-insensitive variant tends to output increasingly large DFAs that obtain better training set accuracy (data not shown), but there is no marked improvement in test set accuracy. Thus the added fitness component that favors smaller DFAs does confer some performance benefit by indirectly selecting for DFAs that can generalize beyond the training set better. Second, it is noted that the size of the best candidate model increases and decreases over the inference process in the active variant of the EEA. It has been found that models that solve all training data so far are gradually replaced by equally-fit but smaller models,

3. In the work reported here two models are output: the best model from each of the two sub-populations.

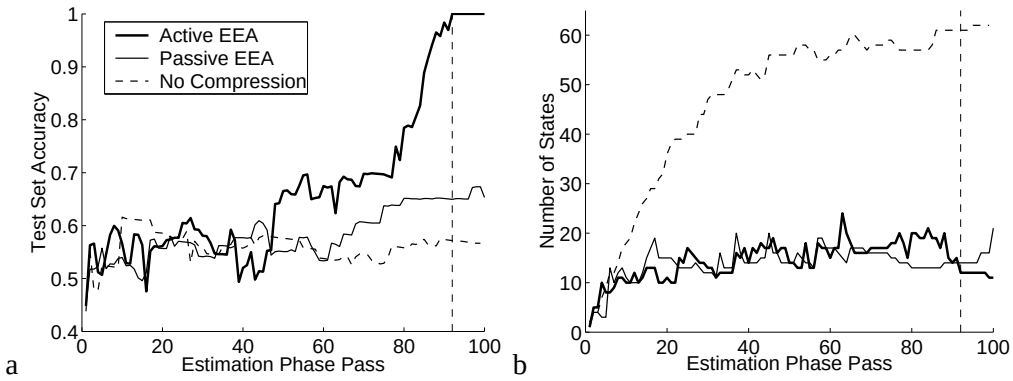

Figure 1: **Performance and Size of Candidate Models. a:** The test set accuracies of the best models output by each pass through the estimation phase. **b:** The total number of states used by each model when labelling the test sentences.

and new training data injected during the inference process causes older, smaller models to fail, only to be replaced with more accurate, larger models. This compression and expansion is a dynamic process that occurs as new training data is collected. Once a model consistent with all training data is obtained by the active EEA, subsequent passes through the estimation phase cause a compression of the candidate model (as long as the model is also consistent with the new training data): the model with $n = 10$ is reduced to $n = 9$ during the 99th pass through the estimation phase for the active EEA variant.

This example illustrates that evolutionary techniques are conducive to dynamic modelling in machine learning, at least in the specific case of grammatical inference, as they allow for dynamic restructuring of both the size of the model (the number of states) and its structure (the connections between states and whether a state is accepting or rejecting) as new training data is collected using active learning.

3.1.2 COMPARATIVE PERFORMANCE AMONG INFERENCE ALGORITHMS

Each of four algorithms—EDSM, Lucas' method, and the active and passive variants of the EEA—were run against 3720 target DFAs: 1200 with $n = 4$, 1200 with $n = 8$, 1200 with $n = 16$, and 120 with $n = 32$ states. For the three smaller DFA classes, each algorithm was applied 100 times for each of 12 training set densities against a different DFA. For the large $n = 32$ DFA class, each algorithm was applied 10 times for each of 12 training set densities to a different DFA due to slower run times on these large DFAs.

The total number of training sentences available for inference for each DFA size and training set density is shown in Table 3.1.2. Table 3.1.2 reports the number of model labellings performed for each run of Lucas' algorithm (left-hand figures) and the EEA (parenthesized figures).

Figure 2 reports the average performance of all four algorithms against the four DFAs and 12 training set densities. Performance is considered to be the test set accuracy of the best DFA output by each algorithm. The test set is comprised of all of the binary sentences that were not selected as training data. In the case of the estimation-exploration algorithm, which outputs two candidate

$d \setminus n$	4	8	16	32
0.01	2	10	40	163
0.02	5	20	81	327
0.03	7	30	122	491
0.04	10	40	163	655
0.05	12	51	204	819
0.06	15	61	245	982
0.07	17	71	286	1146
0.08	20	81	327	1310
0.09	22	91	368	1474
0.10	25	102	409	1638
0.15	38	153	614	2457
0.2	50	204	818	3276

Table 2: Total Numbers of Target Labellings

$d \setminus n$	4		8		16		32	
0.01	0.002	(0.002)	0.010	(0.010)	0.040	(0.040)	0.163	(0.162)
0.02	0.005	(0.005)	0.020	(0.019)	0.081	(0.080)	0.327	(0.321)
0.03	0.007	(0.007)	0.030	(0.029)	0.122	(0.121)	0.491	(0.483)
0.04	0.010	(0.010)	0.040	(0.040)	0.163	(0.162)	0.655	(0.653)
0.05	0.012	(0.012)	0.051	(0.050)	0.204	(0.200)	0.819	(0.810)
0.06	0.015	(0.015)	0.061	(0.060)	0.245	(0.242)	0.982	(0.981)
0.07	0.017	(0.017)	0.071	(0.070)	0.286	(0.279)	1.146	(1.108)
0.08	0.020	(0.019)	0.081	(0.080)	0.327	(0.321)	1.310	(1.243)
0.09	0.022	(0.022)	0.091	(0.090)	0.368	(0.365)	1.474	(1.412)
0.10	0.025	(0.024)	0.102	(0.100)	0.409	(0.403)	1.638	(1.555)
0.15	0.038	(0.037)	0.153	(0.151)	0.614	(0.595)	2.457	(2.371)
0.2	0.050	(0.049)	0.204	(0.200)	0.818	(0.808)	3.276	(3.095)

Table 3: Total Numbers of Model Labellings $(\times 10^9)$

models after the last pass through the estimation phase and then terminates, the best model is taken to be the model from the first sub-population. Figure 3 reports the percentage of runs of both the passive and active EEA variants that produced a model that correctly classifies all training and test data. The percentage of runs that produced such models for the various methods described in Lucas and Reynolds (2005) was not reported.

3.2 Unbalanced DFAs

As can be seen in Figure 2, the EDSM method only begins to compete with the active EEA for DFAs with $n = 32$ states. This seems to suggest that the EDSM methods scale better than the evolutionary method proposed here. However an alternate explanation of this observation is that the EDSM

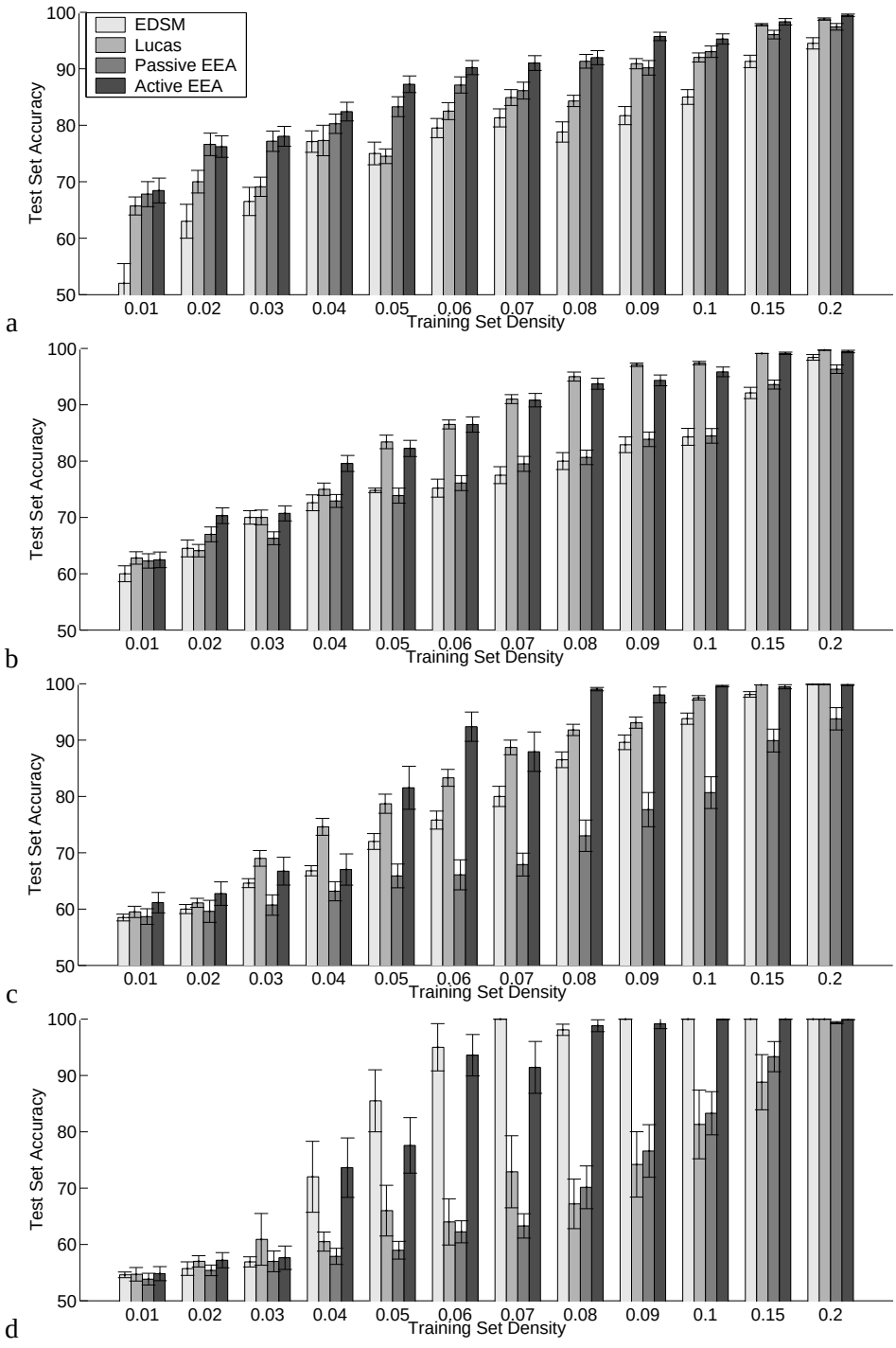

Figure 2: **Comparative Performance against Random DFAs.** The target DFAs are grouped according to size (**a:** $n = 4$, **b:** $n = 8$, **c:** $n = 16$, and **d:** $n = 32$) and training set density. Error bars indicate standard error computed over 100 runs for each algorithm for the first three DFA sizes (**a-c**), and over 10 runs for target DFAs with $n = 32$ (**d**).

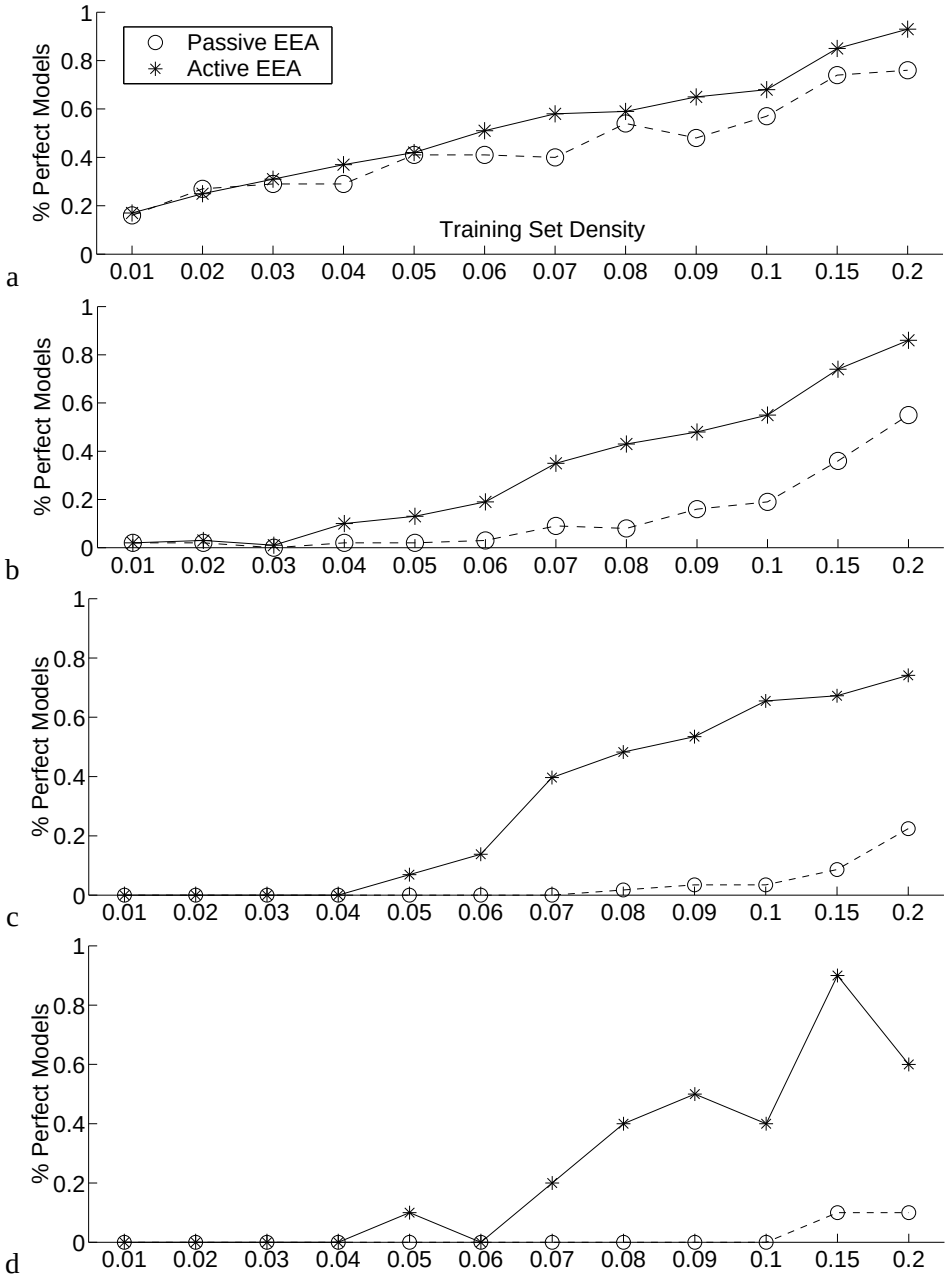

Figure 3: **Probability of Perfectly Consistent Model Discovery for the Random DFAs.** The target DFAs are grouped according to size (**a:** $n = 4$, **b:** $n = 8$, **c:** $n = 16$, and **d:** $n = 32$) and training set density. Data points indicate the percentage of runs that produced a model that achieves 100% test set accuracy.

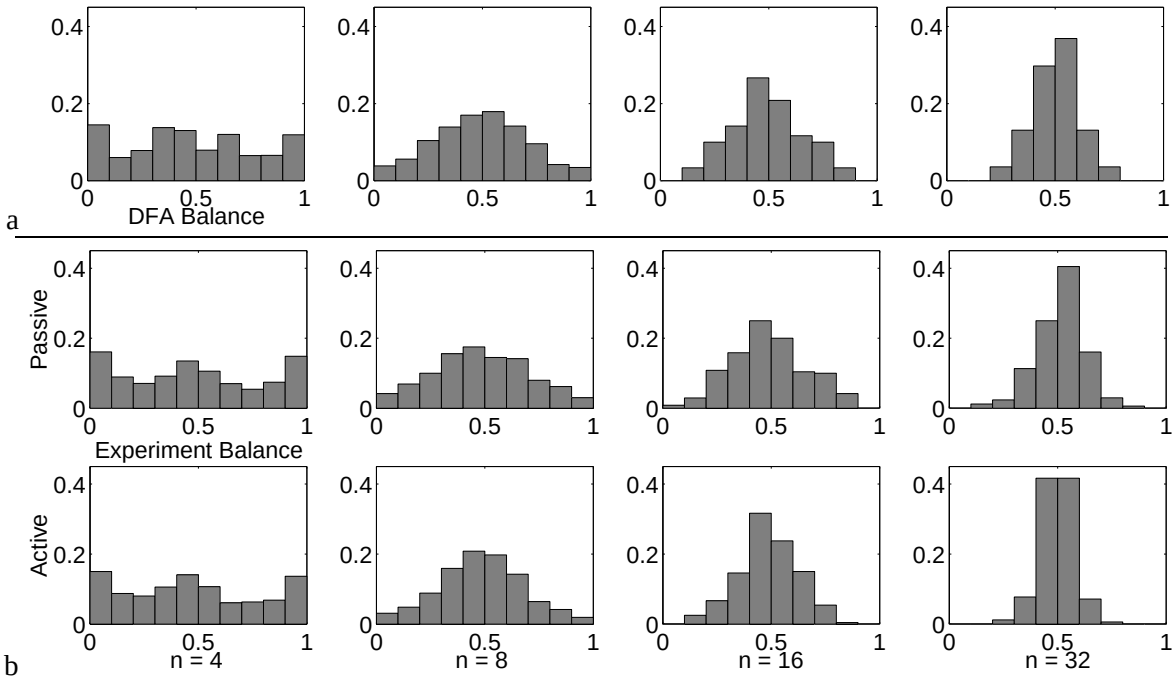

Figure 4: **a: Balance Distribution for the Random DFAs.** The 4 sets of 1200 random DFAs reported in Section 3.1 were grouped according to their size and balance. Balances were calculated using all random strings of length 0 to $\lfloor (2\log_2 n) + 3 \rfloor$. Each bar indicates the fraction of DFAs that fall within that particular range of balances. The upper row of panel *b* shows the distribution of balances for the same set of DFAs. Balances were calculated only using the strings output by the passive variant of the estimation-exploration algorithm applied to that DFA. The bottom row reports the balance distributions of the same DFAs using the active variant of the algorithm.

method works increasingly well on the large instances of the class of random DFAs produced by the generative method proposed by (Lang et al., 1998) and described in Section 3.1.

Figure 4a reports the balance distributions of the four DFA sizes produced by this generative method. As the figure indicates, DFAs with $n = 4$ states tend to exhibit a uniform distribution of balances, but as the DFAs increase in size the distribution clusters more closely around balanced DFAs that produce a more or less equal distribution of positive and negative labellings. For the largest class of DFAs ($n = 32$), the majority of DFAs have a balance within 0.4 and 0.6; the minority of DFAs produce less than 40% labellings of their minority classification.

This agrees with the original stated purpose of this generative approach, which was to produce random, balanced DFAs. However, this raises the possibility that methods developed to infer the DFAs produced by this method may not perform well on other kinds of DFAs, such as unbalanced DFAs. In order to test this, we generalized the generative method to produce DFAs of a desired size and balance. We then generated a number of DFAs of differing sizes and balances, and compared our algorithm against the EDSM algorithm described in Section 2.2. The new generalized method is as follows:

1. Select the desired number of states, n, and the desired balance b. The balance must be a real number in $[0,1]$.
2. Create all random binary strings from length 0 to $\lfloor(2\log_2 n)+3\rfloor$.
3. Create a random digraph with $5n/4$ nodes.
4. While the depth of the graph is not $2\log_2 n-2$, go to step 3.
5. For each state, label it as accepting with probability b; otherwise, label it rejecting.
6. Pass each random string through the DFA, and compute the fraction of positive labellings. If the fraction of positive labellings is not in $[b-\varepsilon, b+\varepsilon]$, go to step 5. ε is taken to be some small tolerance; in the results reported below, ε is set to 0.01.

This method will produce DFAs with size centered around n states, and with a balance in $[b-\varepsilon, b+\varepsilon]$.

Using this method, a total of 15 DFAs were created: 5 with $n=8$ states, 5 with $n=16$ states, and 5 with $n=32$ states. Within each size class, five DFAs were created with balances of 0.1, 0.2, 0.3, 0.4 and 0.5. For each DFA, the EDSM and the active variant of the EEA were applied 30 times using 12 different training set densities. The EDSM and EEA were instantiated using the same parameters described in the previous section.

Due to speed limitations, the EEA was not applied to the $n=32$ DFAs using training set densities above 0.06. The mean test set accuracies of the EDSM and EEA methods are reported for the $n=8$ DFAs in Figure 5, for the $n=16$ DFAs in Figure 6, and for the $n=32$ DFAs in Figure 7. The improvement factor for each DFA and corresponding training set density was calculated using $m_{\text{EEA}}/m_{\text{EDSM}}$, where m_{EEA} is the mean test set accuracy of the EEA for that DFA and that training set density, and m_{EDSM} is the mean accuracy for the same DFA and same training set density.

4. Discussion

Several trends can be noted in the mean performances of the algorithms reported in Figure 2. First, the evolutionary approach of Lucas and Reynolds (2005) tends to outperform the EDSM variant for smaller target DFAS ($n<32$), but the EDSM far outperforms Lucas' algorithm for larger target DFAs ($n=32$). Second, the active EEA variant outperforms all three other algorithms for the smallest size of target DFA ($n=4$); is competitive with Lucas' algorithm for DFAs with $n=8$ and $n=16$; and is competitive with EDSM on the largest target DFAs ($n=32$).

Third, the passive EEA variant performs poorly against the other three algorithms on all larger DFAS ($n>4$). Because the passive variant performs the same or less model evaluations than Lucas' algorithm, and it randomly selects the same number of training sentences, we can conclude that our particular method of evolutionary search is inferior to that proposed by Lucas and Reynolds (2005). It seems plausible that replacing the evolutionary search that occurs within the EEA with a more powerful search technique—whether another evolutionary method, or a heuristic variant such as EDSM—may allow the proposed algorithm to outperform the passive forms of grammatical inference reviewed here.

The reason that the EDSM begins to compete with the EEA on the large $n=32$ DFAs is made clear in Figure 7. The EDSM only performs well on DFAs centered at $b=0.4$ (indicated by the

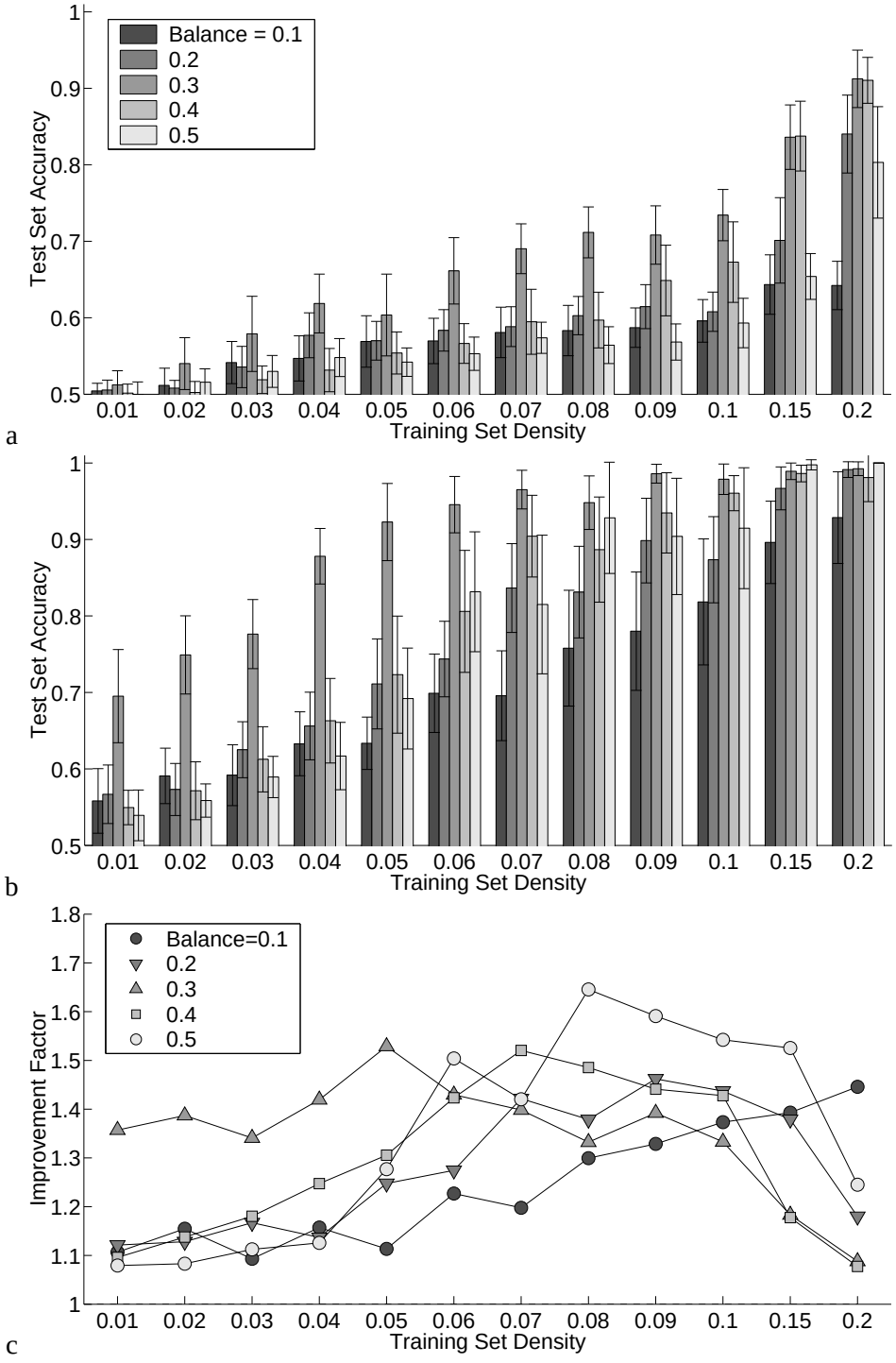

Figure 5: **Performance of the EEA against EDSM on DFAs with $n = 8$ states and differing balances. a:** Mean accuracies of the EDSM for differing balances and training set densities. **b:** Mean accuracies of the EEA for differing balances and training set densities. **c:** Improvement factors of the EEA over the EDSM for differing balances and training set densities.

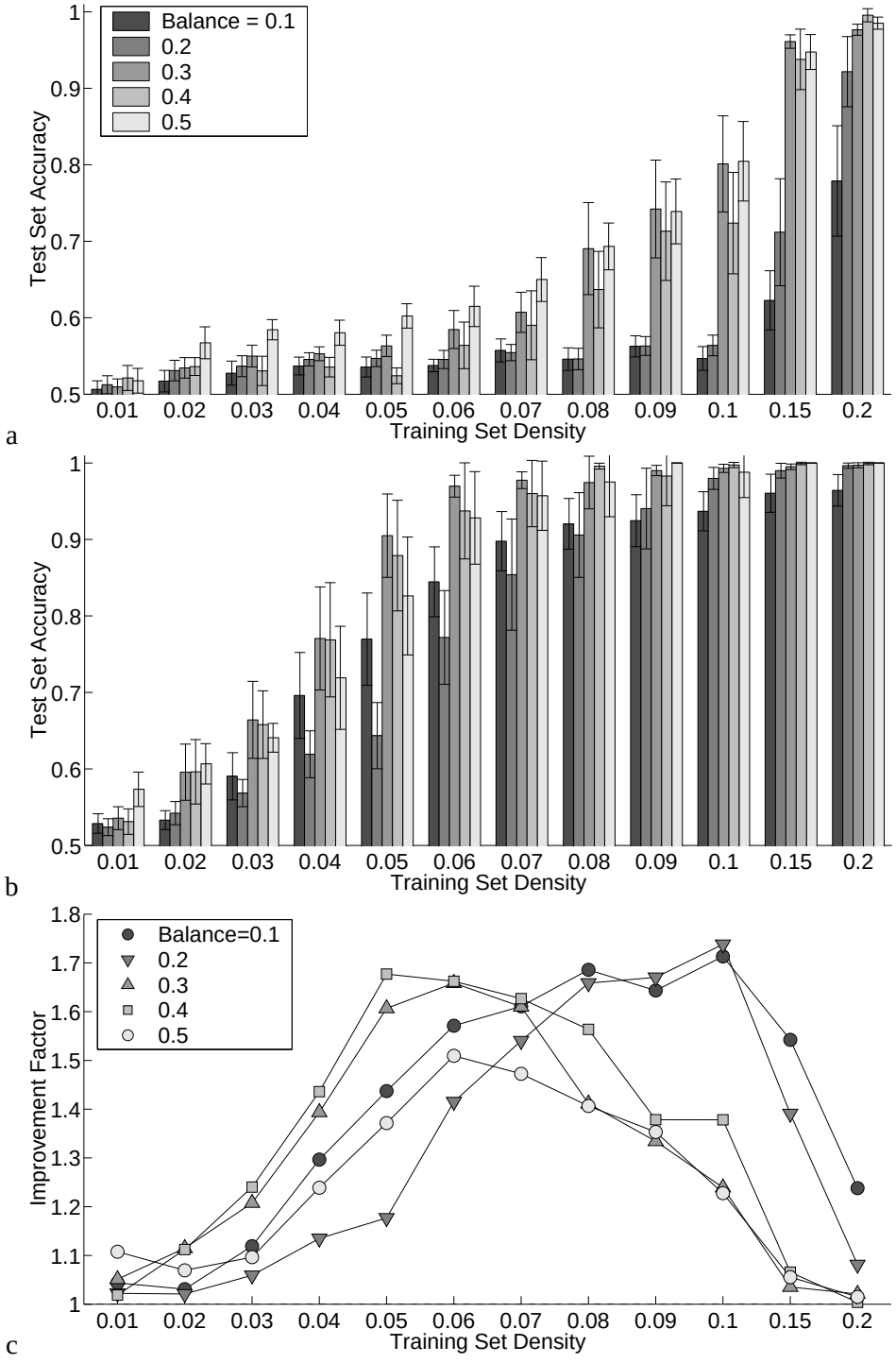

Figure 6: **Performance of the EEA against EDSM on DFAs with $n = 16$ states and differing balances. a:** Mean accuracies of the EDSM for differing balances and training set densities. **b:** Mean accuracies of the EEA for differing balances and training set densities. **c:** Improvement factors of the EEA over the EDSM for differing balances and training set densities.

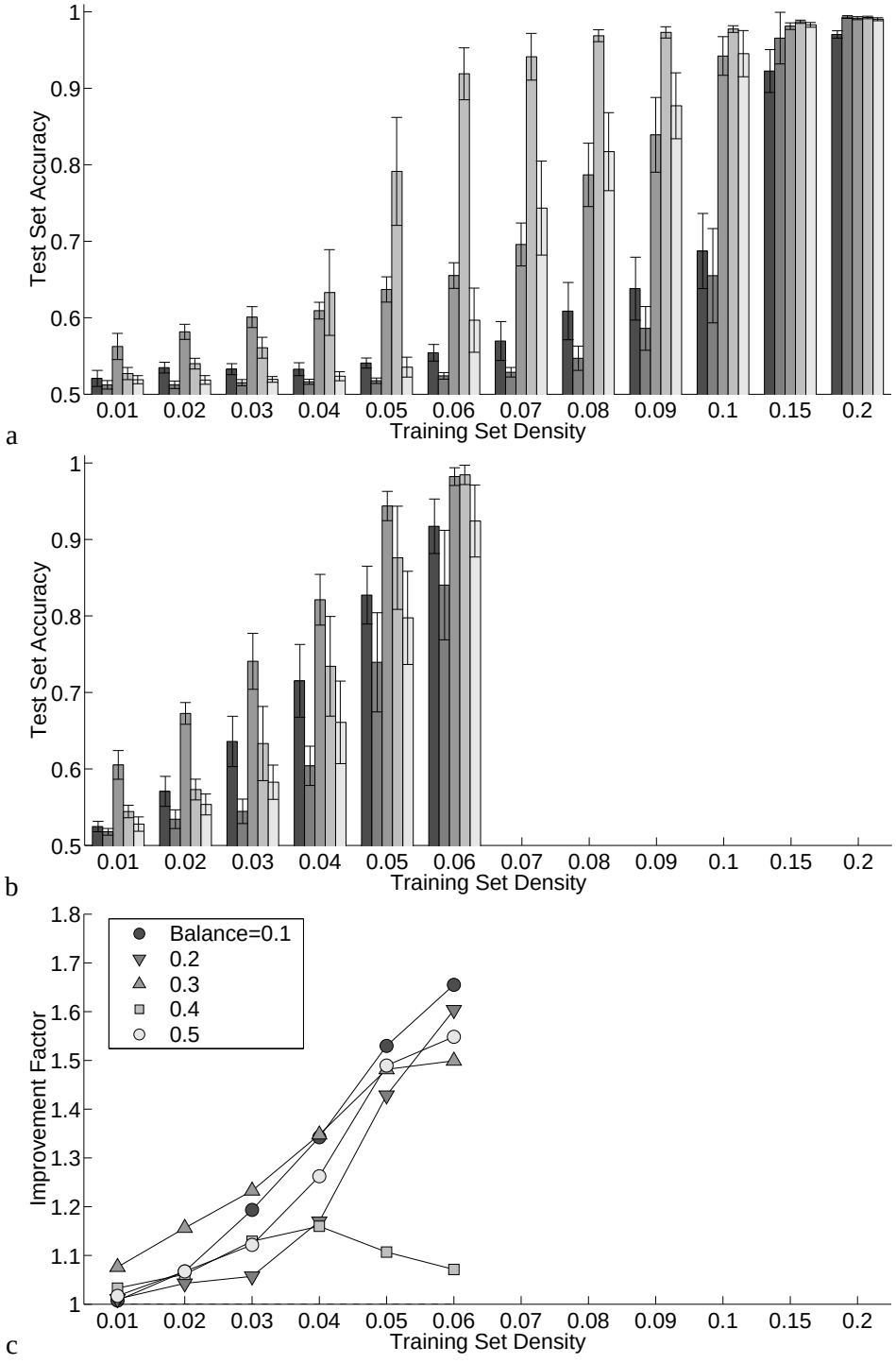

Figure 7: **Performance of the EEA against EDSM on DFAs with $n = 32$ states and differing balances. a:** Mean accuracies of the EDSM for differing balances and training set densities. **b:** Mean accuracies of the EEA for differing balances and training set densities. **c:** Improvement factors of the EEA over the EDSM for differing balances and training set densities.

balance$=0.4$ bins in Figure 7a), and only slightly worse on DFAs with balances centered at $b=0.5$ (indicated by the balance$=0.5$ bins) for the intermediate training set densities. For low training set densities it does poorly on the DFAs of all balances, and the highest training set density it does well on DFAs of all balances. Finally, on the unbalanced DFAs centered around 0.1 and 0.2, it only begins to infer accurate models when supplied with training set densities of 0.15 and 0.2.

As Figure 4a illustrates, the generative method proposed by Lang et al. (1998) mostly produces random DFAs with balances in $[0.4,0.6]$, which corresponds to the range of DFA balances for which the EDSM is well-suited. In contrast, as the EEA begins to perform successfully on the DFAs when allowed to generate sufficient training data, it begins to perform successfully on all of the DFAs, regardless of balance. As can be seen in Figure 7b, the standard deviations of the test set accuracies mostly overlap within each training set density class for the EEA, while the deviations of accuracies for the EDSM in Figure 7b do not overlap in many cases. This difference is highlighted by Figure 7c, which shows that for training set densities above 0.04, the EEA performs significantly better than the EDSM for all DFA balances except 0.4. From this it can be concluded that at least this EDSM variant performs well on just those DFAs with balances equal to those produced by the generative method proposed by Lang et al. (1998).

The balance specificity of the EDSM method is also clear in Figure 6a: EDSM does well for DFAs with balances of 0.3 and higher, but requires high training set densities to perform well on the DFAs with balances below 0.3. Alternatively, the EEA begins to perform better on all DFA balances as training set density increases (Figure 6b). Again this difference is highlighted by Figure 6c, which shows that for training set densities above 0.08, there is a significant performance increase of the EEA over the EDSM for the two imbalanced DFAs with $b=0.1$ and $b=0.2$.

In Figure 5, the balance specificity of the EDSM is less apparent. However, Figure 5c indicates that for DFAs with $b=0.1$, the EEA achieves an increasing performance benefit over the EDSM for increasing training set densities (indicated by the increasing slope of the line with darkened circle markers). For instance, the EDSM only achieves a mean test set accuracy of 62% for the DFA with $n=8$ and $b=0.1$ using 0.2 training set density, while the EEA achieves a mean accuracy of 93% for the same DFA and the same amount of training data.

The reason why the active EEA infers imbalanced DFAs better than the EDSM is made clear by Figure 4b. For DFAs with $n=32$ states, it can be seen that the active EEA generates training strings that achieve a more balanced labelling (lower righthand panel) than the passive EEA variant which outputs random strings for labelling (middle righthand panel). This is explained as follows. At the outset of inference using the active EEA, training strings are generated at random, because sufficiently accurate models do not yet exist. As inference proceeds, a few training strings are output to the target system that obtain a minority labelling. This allows for an increase in the accuracy of the set of candidate models in the estimation phase. Henceforth, training sets are evolved that cause disagreement among the models. This indicates that training strings with high fitness at least elicit a minority labelling from some of the candidate models, and since the models are now somewhat accurate, this increases the probability of obtaining a new minority labelling from the target system. As inference proceeds, minority labellings are extracted with increasing probability, allowing for the better inference of imbalanced DFAs.

This advantage of active training data generation, compared to passive training data collection, is also supported by the results reported in Figure 3. Clearly, the active EEA discovers models consistent with all training and test data more often than the passive EEA.

These results highlight the need to broaden the class of DFAs that are considered in grammatical inference. Furthermore, an inference algorithm should be judged not just on how well it infers a DFA of a given size and given training data, but how well an algorithm does on DFAs of differing sizes and balances. This requires rethinking how grammatical inference algorithms are compared: rather than simply providing them with training data already collected from a DFA, the algorithms should be free to request training data labelling, as is done in the active learning community (Baram et al., 2004).

4.1 Intelligent Testing

The results reported here support the claim that useful experiments (in the domain of grammatical inference, binary sentences) are those that elicit informative responses from the target system. What qualifies as 'informative', however, will vary across problem domains. Here we have stressed that one informative type of test are training data belonging to the minority class of an imbalanced DFA: automatically and actively generating such informative tests helps the algorithm to outperform other methods that rely on passively generated random training data.

It is important to note that there is no explicit reward in the exploration phase for such sentences. Rather, the ability to cause disagreement between alternative, approximate models means that $k/2$ of the models will yield the minority label for such a sample, and because these models are somewhat accurate (but not yet perfect), there is an increased probability that that sentence will actually elicit the minority label from the target DFA. This is a useful trait to have in an inference method, because what qualifies as an informative is domain dependent. For example in the domain of classification, training data that lie near the intersection of the decision boundaries of candidate classifiers would be more informative than data that lies on the same side of the decision boundaries.

There may be other kinds of informative sentences in grammatical inference that are unknowingly being favored by the active EEA variant. For example it seems plausible that for many DFAs, longer sentences are more informative than shorter sentences. It is clear that states closer[4] to the start state in the transition function T will be visited more often than distant states, and longer sentences have a higher probability of reaching these distant states than shorter sentences do. Also, because longer sentences traverse more state transitions than shorter sentences and therefore have a better chance of uncovering differing transitions among candidate models, we predict that longer strings would tend to produce more disagreement among candidate models than shorter sentences can. So, we predict that the active EEA variant will propose, on average, longer training sentences than a passive algorithm will. Whether longer sentences truly are more informative than shorter ones, and whether longer sentences are actually favored by the active EEA variant has not yet been verified.

Cast in another light, informative tests tend to expose the unobservable parts of the target system, thus accelerating the inference process. In grammatical inference, unbalanced DFAs are less observable than balanced DFAs: there are either less states that produce the minority labelling than states that produce the majority labelling, or minority labelling states are more distant from the start state than majority labelling states. It follows then that passive grammatical inference approaches are inappropriate in these cases, for one of two reasons: either a balanced training set is assumed, in which case the minority class is grossly over-represented in the training data; or random training

4. Here, we assume that distance between states—more specifically, the distance between the start state and a given state—is viewed as the number of paths that exist between those states, and the mean length of those paths.

data is assumed, in which case the minority class is grossly under-represented. An active approach to grammatical inference, like the estimation-exploration algorithm, actively generates a training set that falls between these two extremes, and accelerates inference.

It also seems clear that most real-world systems will be unbalanced: they will not produce equal numbers of all label types for a randomly generated set of sample data. Also, acquiring a balanced training set will require a large number of target labellings in order to obtain enough of the minority labellings. As stated previously, the estimation-exploration algorithm is designed for inference using as few target trials as possible, because real world systems may be costly, dangerous or slow to query.

Furthermore, our method may be useful for indicating what kinds of training data is most useful for the inference of particular kinds of languages, by simply observing what kinds of sentences are generated by our method. However, this line of investigation has not yet been pursued.

4.2 Time Complexity

The running time of the estimation-exploration algorithm is proportional to the total number of labellings (trials) performed by the target DFA (t):

$$T(t) = gp\sum_{i=1}^{t} i + gtp^2 \quad (8)$$

$$= O(t^2), \quad (9)$$

where p and g are the population size and number of generations used by the genetic algorithm during each pass through either phases, respectively. The first term accounts for the running time of the estimation phase, which evolves models against all labellings seen so far. The second term accounts for the running time of the exploration phase, which evolves candidate trials and evaluates each one against each individual in the population of models, to estimate overall disagreement.

However, in the implementation of the EEA described here, a trial is not evaluated against all of the candidate models; a trial is only evaluated against the two best models from each of the two model sub-populations. This reduces overall running time but does not affect the time complexity of the algorithm:

$$T(t) = gp\sum_{i=1}^{t} i + 2gtp \quad (10)$$

$$= O(t^2). \quad (11)$$

Therefore, the algorithm running time increases polynomially with the total number of training strings presented to the target DFA.

However, the completion of the algorithm does not guarantee the output of a model that can correctly classify all training and test strings. Due to the complexity of the learning algorithm and its stochastic nature, we have not yet characterized the time complexity required to guarantee the output of such a consistent model. However, Figure 3 provides empirical evidence that for the random DFAs generated using the method proposed by Lang et al. (1998), a model DFA that correctly classifies all binary strings with lengths of 0 to $\lfloor(2\log_2 n)+3\rfloor$ can be found using the proposed algorithm. More specifically, for the active EEA, a model consistent with all training and test strings was found in at least 1 of the 100 runs for all training set densities for the $n = 4$ target

DFAs (Figure 3a); for all training set densities of 0.04 or higher for the $n = 8$ target DFAs (Figure 3b); for all training set densities of 0.05 or higher for the $n = 16$ target DFAs (Figure 3c); and for all training set densities of 0.07 or higher for the $n = 32$ target DFAs (Figure 3d). This translates to the ability of the algorithm to find perfectly consistent models when it is allowed to propose at least 2 binary strings for labelling by a $n = 4$ target DFA; at least 40 binary strings for labelling by a $n = 8$ target DFA; at least 204 binary strings for labelling by a $n = 16$ target DFA; and at least 1146 binary strings for labelling by a $n = 32$ target DFA (data taken from Table 3.1.2). This indicates that the ability of the active EEA to produce a model consistent with all training and test data scales polynomially with the amount of training data. However, as reported in Section 3.2, the balance of a DFA also has an effect on inference ability for both EDSM and EEA methods. Formulating time complexities for both methods as a function of both DFA size and balance as well as allowed number of target labellings requires further investigation.

5. Summary and Conclusions

Here we have introduced a co-evolutionary approach to grammatical inference that differs from both passive and active learning methods in that training data is not assumed to be provided by an external agent (either all at once or iteratively), but is generated internally by the algorithm itself: one component of the algorithm evolves a pool of candidate models, and the second component evolves a new training sentence that causes maximal disagreement among them. The sample that causes the most disagreement is then sent to the target language for labelling. We refer to this method as the estimation-exploration algorithm, or EEA. We have previously used this method to infer other kinds of tightly coupled, nonlinear target systems (see Bongard and Lipson, 2005, for an overview).

It has been shown here that the EEA outperforms another evolutionary approach to random DFA inference. Furthermore, the EEA outperforms the more powerful of the heuristic approaches (EDSM) on small random DFAs, and is competitive with EDSM on larger random DFAs.

The reason why the EDSM methods seem to perform better as the target DFAs increase in size was investigated. It was found that the EDSM method investigated here does not improve in ability as DFAs increase in size, but rather performs well on DFAs with specific balances (percentages of positive labellings), and that the generative method introduced by Lang et al. (1998) produces large DFAs with just these balances. It was shown that the EEA performs better on DFAs with differing balances by actively extracting minority labellings from the target DFA.

In order to better gauge the inference ability of grammatical inference methods, we introduce a more general method for generating DFAs with specific sizes and balances. In future, methods should be shown to work well not only on large DFAs with limited training data, but also consistently on DFAs of the same size but differing balances.

Our algorithm also allows for continual expansion and compression of candidate models over time, in response to new training data: expansion allows for the accommodation of new training data, and compression usually leads to greater test set accuracy. The current EDSM methods only allow for state compression, raising the possibility of inaccurate generalization. Another benefit of the EEA is that the internal search mechanism could be replaced with a more powerful search method: our algorithm functions independently of the search method used for inferring models and generating informative tests. It may be that replacing the current, basic evolutionary method with a more powerful stochastic search method (or even a deterministic one such as an EDSM variant) may

improve our method further. At the moment an upper bound is currently assumed on the number of states in a candidate model, but again, the current model search mechanism in EEA could be replaced by one that does not assume an upper bound, as was done in Luke et al. (1999).

In many approaches to grammatical inference, either a balanced training set is assumed, or random training data is generated. This can be wasteful when inference should be performed with as few labellings by the target language as possible, because either too little or too much of the minority training class is passively collected, or many labellings have to be performed in order to collect enough of the minority class training data for a balanced set. This is of practical importance because for many real-world languages or classifiers, collection of training data can be costly, dangerous or slow. In the EEA, only training sentences that cause maximal disagreement among the current set of candidate models are sent to the target DFA for labelling, and here we have shown that this process builds an informative training set: the set contains sufficient minority class training data to produce accurate models.

In future work we intend to apply our algorithm to probabilistic finite automata—automata that output probabilities as to which class(es) a sequence may belong, rather than absolute class assignments—as well as noisy sample data: evolutionary methods have previously proven to be well suited to dealing with probabilistic and noisy systems. One possible approach would be to evolve test sequences that cause the candidate models to disagree most in the class probabilities they predict the target system will output for that sequence. We also intend to apply our method to larger languages ($n >> 32$) in order to provide evidence that our approach could be useful in real-world situations.

In closing we suggest that the grammatical inference community consider broadening the suite of target systems and target system generation methods in order to avoid biasing the development of new inference methods that only perform well on the target systems produced by a particular generative method.

Acknowledgments

This work was supported by the National Academies Keck Futures Grant for interdisciplinary research, number NAKFI/SIG07. 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.

References

D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

D. Angluin. Queries revisited. *Theoretical Computer Science*, 313:175–194, 2004.

Y. Baram, R. E. Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

T. Berg, B. Jonsson, M. Leucker, and M. Saksena. Insights to Angluin's learning. Technical Report 2003-039, Uppsala Universitet, 2003.

F. Bergadano and D. Gunetti. *Inductive Logic Programming: From Machine Learning to Software Engineering*. MIT Press, Cambridge, MA, 1995.

J. Bongard and H. Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, pages 169–176. IEEE Computer Society, 2004a.

J. Bongard and H. Lipson. Automating genetic network inference with minimal physical experimentation using coevolution. In *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO)*, pages 333–345. Springer, 2004b.

J. Bongard and H. Lipson. Automating system identification using co-evolution. *IEEE Transactions on Evolutionary Computation*, 9(4):361–384, 2005.

S. Brave. Evolving deterministic finite automata using cellular encoding. In *Genetic Programming 96: Proceedings of the First Annual Conference on Genetic Programming*, pages 39–44. MIT Press, 1996.

O. Cicchello and S. C. Kremer. Inducing grammars from sparse data sets: A survey of algorithms and results. *Journal of Machine Learning Research*, 4:603–632, 2003.

P. Dupont. Incremental regular inference. In L. Miclet and C. Higuera, editors, *Proceedings of the Third ICGI-96, Lecture Notes in Artificial Intelligence 1147*, pages 222–237, 1996.

P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI'94)*, pages 25–37, 1994.

K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In *Grammatical Inference (Lecture Notes in Artificial Intelligence 1433)*, pages 1–12. Springer-Verlag, 1992.

K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In V. G. Honavar and G. Slutzki, editors, *Proceedings of the Fourth International Colloquium on Grammatical Inference: ICGI 1998*, pages 1–12, London, UK, 1998. Springer-Verlag.

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

S. M. Lucas and T. J. Reynolds. Learning deterministic finite automata with a smart state labelling evolutionary algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27: 1063–1074, 2005.

S. Luke, S. Hamahashi, and H. Kitano. "Genetic" programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1098–1105. Morgan Kaufmann, 13-17 1999.

S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Urbana, IL, USA, 1995.

J. Oncina and P. Garciá. Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, pages 49–61. World Scientific, 1992.

T. Pao and J. Carr. A solution of the syntactic induction-inference problem for regular languages. *Computer Languages*, 3:53–64, 1978.

R. G. Parekh and V. G. Honavar. Efficient learning of regular languages using teacher supplied positive examples and learner generated queries. In *Proceedings of the Fifth UNB Conference on AI*, pages 195–203, 1993.

R. G. Parekh and V. G. Honavar. An incremental interactive approach for regular grammar inference. In *Proceedings of the Third ICGI-96 (Lecture Notes in Artificial Intelligence 1147)*, pages 238–250. Springer Verlag, 1996.

L. Pitt. Inductive inference, DFAs and computational complexity. In *Proceedings of the International Workshop on Analogical and Inductive Inference (Lecture Notes in Artificial Intelligence 397)*, pages 18–44. Springer Verlag, 1989.

S. Porat and J. Feldman. Learning automata from ordered examples. *Machine Learning*, 7:109–138, 1991.

J. M. Sempere and P. Garcia. A new regular language learning algorithm from lexicographically ordered complete samples. In *IEEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives*, pages 6/1–6/7, 1993.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, pages 387–294, San Mateo, CA, 1992. Morgan Kauffman.

M. Tomita. Dynamic construction of finite automata from examples using hill climbing. In V. G. Honavar and G. Slutzki, editors, *Proceedings of the Fourth Annual Cognitive Science Conference*, pages 105–108, 1982.

B. A. Trakhtenbrot and Y. M. Barzdin. *Finite Automata Behavior and Synthesis*. North-Holland Publishing Company, Amsterdam, 1973.