Learning Cassin's Vireo (*Vireo cassinii*) syntax through grammatical inference

Julio G. Arriaga[1], Richard Hedley[2], Edgar E. Vallejo[1] and Charles E. Taylor[2]

[1]Computer Science Dept., Tecnológico de Monterrey, Campus Estado de México
Atizapán de Zaragoza, Estado de México, 52926, México
[2]Dept. of Ecology and Evolutionary Biology, University of California, Los Angeles
Los Angeles, CA, 90095-1606, USA
elbuenjulius@gmail.com

Abstract

Birdsong may be regarded as a complex adaptive system. In this paper we study the relationship between complexity and consistency of an evolving model of Cassin's Vireo syntax, using a genetic algorithm to approximate a Minimal Consistent Deterministic Finite-state Automata (MCDFA) capable of accepting vocal sequences produced by birds of the study species. Our results imply that, despite the complex vocal behaviour of this species, the complexity of the model can be reduced considerably to encompass all of the positive samples while retaining the ability to exclude similar negative samples. These results suggest the existence of important regularities in the song sequences of this species.

Introduction

The study of birdsong as a science dates back to the 1950s when the first sound spectrographs became available, enabling researchers to objectively study and specify its structure with an unprecedent level of detail. Since then, much research have focused on deciphering its purpose, learning mechanisms, and meaning behind them.

There have been considerable efforts in the past towards understanding the structural rules that govern birdsong production (Honda and Okanoya, 1999; Berwick et al., 2011). However, due to the vast diversity of songbird species and the apparent structural differences in the songs they produce there is still no consensus on the limits of their complexity.

Similarly, birdsong have proven to be an excellent platform for exploring a variety of topics that are relevant to artificial life research, such as complexity, coevolution and sexual selection, among others (Taylor and Cody, 2015; Sasahara and Ikegami, 2003, 2004).

Our research is currently focused on the acoustic monitoring of different species of birds in several areas of the US and Mexico where they are abundant (Arriaga et al., 2013). Our long term goal is to understand the structure and function of birdsong. Particularly, the research described here aims at analysing the trade-off between complexity and consistency when learning a model representation of the syntactic rules governing birdsong structure of Cassin's Vireo (*Vireo cassinii*), a songbird possessing a complex vocal behaviour.

More precisely, we explore how the complexity of the model describing the syntax of CaVi song could be reduced while avoiding overgeneralisation. Toward that goal, we use a genetic algorithm to approximate a Minimal Consistent Deterministic Finite-state Automata (MCDFA) capable of accepting sequences produced by birds of the Cassin's Vireo species and rejecting a collection of artificially generated negative sequences.

The problem of finding a MCDFA from a set of examples S is $NP - Complete$ (Gold, 1978), further even finding an approximately small consistent DFA has also been proven to be not solvable in polynomial time (Pitt and Warmuth, 1989). Given these constraints, approaching the problem as an optimisation task using evolutionary computation is a natural approach as exhaustive methods are not tractable.

Previous research on the area by (Kakishita et al., 2009) has already tackled the problem of finding a minimised automata representation from observed song sequences, however this approach assumes the target representation is a k-reversible language. Although this assumption is backed by some biological plausibility, we consider exploring methods not bound by these a priori assumptions an interesting line of research.

Materials and Methods

Study Species

Cassin's Vireo (*CaVi*) is a small migratory songbird that breeds throughout western North America during the April-July period. Individuals are territorial, establishing and defending their breeding grounds from other males. Singing is exclusive to males and is primarily used during the breeding season (Goguen and Curson, 2002). In our studied population each male possesses a repertoire comprised of 51 ± 5 (mean $\pm$ standard deviation) highly stereotyped phrase-types, each lasting between 0.20 and 0.66 seconds (Hedley, 2016). Individuals sing persistently throughout the day, delivering phrases at varying rates: from more than one phrase per second during periods of high vocalisation production, to single phrases delivered between silence intervals of several seconds long (Hedley, 2016).

Field Recordings

All recordings were collected on private land five kilometres north of the town of Volcano in Amador County, California (10 S 706584 4262742) by Richard Hedley throughout the breeding season, from April 25 to June 28, 2013 and from May 5 to June 25, 2014. This period was deliberately chosen to account for possible seasonal variations in singing behaviour and to increase the likelihood of observing the majority of the temporal variability of the songs, as song output in the species is concentrated in the breeding season.

Recordings were made using a Marantz PMD 661 solid-state digital recording unit and a Sennheiser MKH20-P48 microphone with a Telinga parabolic reflector. Each recording session began when a bird was heard singing, and ended either when the bird stopped for a significant amount of time or flew away, becoming inaudible. Recordings were thus opportunistic in nature, without clear beginnings or endings in some instances.

Recording Annotation

Using the linguistics program Praat (Boersma and Weenink, 2014), each recording was subsequently annotated, identifying and categorising distinct phrase-types through visual inspection of their spectrograms. A unique two letter code (*aa*, *ab*, *ac*, etc.) was assigned to each phrase-type and a spectrogram image of the phrase-type was added to a reference catalogue for future phrase identification. Figure 1 shows spectrogram representations of three distinct phrase-types, as recorded from two different individuals in our study population.

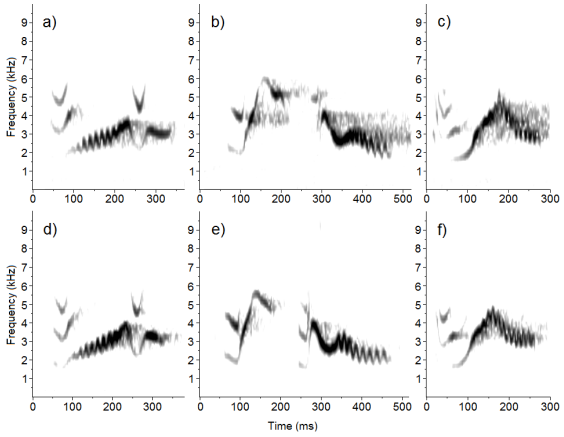

Figure 1: Spectrogram images of three distinct phrase-types, recorded from two different individuals. Panels a) and d) illustrate phrase-type *ai*; panels b) and e) illustrate phrase-type *bj*; and panels c) and f) illustrate phrase-type *br*. The first three exemplars (a-c) were recorded from the *Meadow* individual, while the last three (d-f) were recorded from the *Sign* individual.

Phrases belonging to each phrase-type were remarkably stereotyped, such that each phrase was readily assigned a phrase-type label by visually inspecting a spectrogram of the signal. We determined that this method of phrase identification was objective by subsequent classification with a variety of machine learning methods (Tan et al., 2015; Kantapon et al., 2015); these agreed almost perfectly with the human identification. This method has been found to be more than 99% accurate for annotation of phrase-types for the purpose of syntactic analysis (Hedley, 2016). We identified a total of 128 distinct phrase-types among the 15 different individuals in our study. This group of 15 birds represents the entire male population of *CaVi* individuals living within the 1-square kilometre valley of our study site.

Recordings were also tagged with the designated name of the individual bird which produced it. This was done by the recordist through means of visual identification and territory analysis, and was later verified through the use of an ensemble of Machine Learning Methods (Arriaga et al., 2016). Table 1 shows an example of some recordings as annotated phrase sequences.

Table 1: Example of three recordings as sequences of phrases after being manually annotated. Each two-letter code corresponds to a distinct phrase-type identified within the bird species.

Individual	Annotated Recording
agbk	ah, ai, ah, aj
	ah, ai, en, aj
	fg, em, cg, cr, fq
	ai, en, ai, aj, en, en, ak
	fg, em, ck, fg, ca, fg, em

Data Preparation

Recordings were divided into phrase sequences by grouping phrases sung with no more than ten second pauses between them. All phrase sequences were then grouped by the individual which produced them. Since all our data consists of observed sequences, all sets consist entirely of positive samples (S_+). Negative samples are fundamental to avoid excessive generalisation of the model. For this purpose we artificially generated a negative sample S_{-_1} of phrase sequences by randomly sampling phrases from a collection of all observed phrase-types for each individual. The sequences produced are thus composed of uniformly distributed phrase-types, and phrase-type combinations or *n-grams*. Previous research on the composition of observed sequences has shown non-random patterns of sequential distribution of phrase-types and *n-grams* (Hedley, 2016). Although we cannot guarantee all of these simulated phrase sequences are impossible to produce by the birds' syntax, their statistical composition make them highly unlikely. Ad-

ditionally, a second set of negative examples S_{-_2} was generated by randomly sampling phrases from a collection of all observed phrase-types for each individual maintaining the same phrase-type frequency distribution as the observed data. Figure 2 (a) and (d) show the total number of occurrences of distinct phrase-types and bigrams ordered by rank for the observed data of individual *agbk* compared to those in the simulated data S_{-_1} (b) and (e), and S_{-_2} (c) and (f).

Minimal Consistent Deterministic Finite-State Automata

A Deterministic Finite-State Automata (DFA) is a quintuple $A = \langle \Sigma, Q, q_\lambda, \mathbb{F}, \delta_N \rangle$ where Σ is an alphabet, Q is a finite set of states, $q_\lambda \in Q$ is the initial state, $\mathbb{F}$ denotes the set of final states (both accepting $\mathbb{F}_\mathbb{A}$ and rejecting $\mathbb{F}_\mathbb{R}$), and $\delta_N : Q \times (\Sigma) \rightarrow Q$ is a transition function.

The minimal DFA consistent with a given sample S is simply a DFA with at most *opt* states that accepts all S_+ positive examples and rejects all S_- negative samples. Where *opt* is the least number of states possible for the given sample. Given the hardness of the problem, an exhaustive search of all possible solutions is not practical for most cases.

Genetic Algorithm

A genetic algorithm was used to explore possible models for the observed data for each individual. All experiments were performed in the Python programming language (van Rossum and Drake, 2001), using the NetworkX (Hagberg et al., 2008) and DEAP (Fortin et al., 2012) packages. Positive samples (S_+) were used to build a Maximal Canonical Automata (MCA), a star shaped Non-Deterministic Finite-state Automata (NFA) with one branch for each sequence in S_+.

In other words, a MCA is an automata that accepts exclusively the sequences observed in the data. See Figure 3.

A MCA is also structurally complete with respect to the positive sample, as every transition and acceptor state is used at least once when parsing the observed strings. We thus defined our search space as the subset of all automata consistent with S which are structurally complete. Since we are interested in finding a DFA, non-deterministic transitions were removed from the MCA. The resulting automata is equivalent to a Prefix Tree Acceptor (PTA). See Figure 4.

Population

Individuals were defined as partitions over the PTA in order to explore the search space. A partition Π of a DFA A is defined as $A/\Pi = \left(\Sigma, \bar{Q}, \bar{q_\lambda}, \bar{\mathbb{F}}, \bar{\delta_N}\right)$

- $\bar{Q} = Q/\Pi$ is the set of equivalence classes defined by the partition Π.
- $\bar{\delta_N}$ is a function $\bar{Q} \times \Sigma \rightarrow Q$ such that $\forall \bar{q}, \bar{q}' \in \bar{Q},\ \forall a \in \Sigma, \bar{q}' \in \bar{\delta_N}(\bar{q}, a)\ \ if_{def}\ \exists q \in \bar{q}\ \exists q' \in \bar{q}'\ :\ q' \in \delta_N(q, s)$.

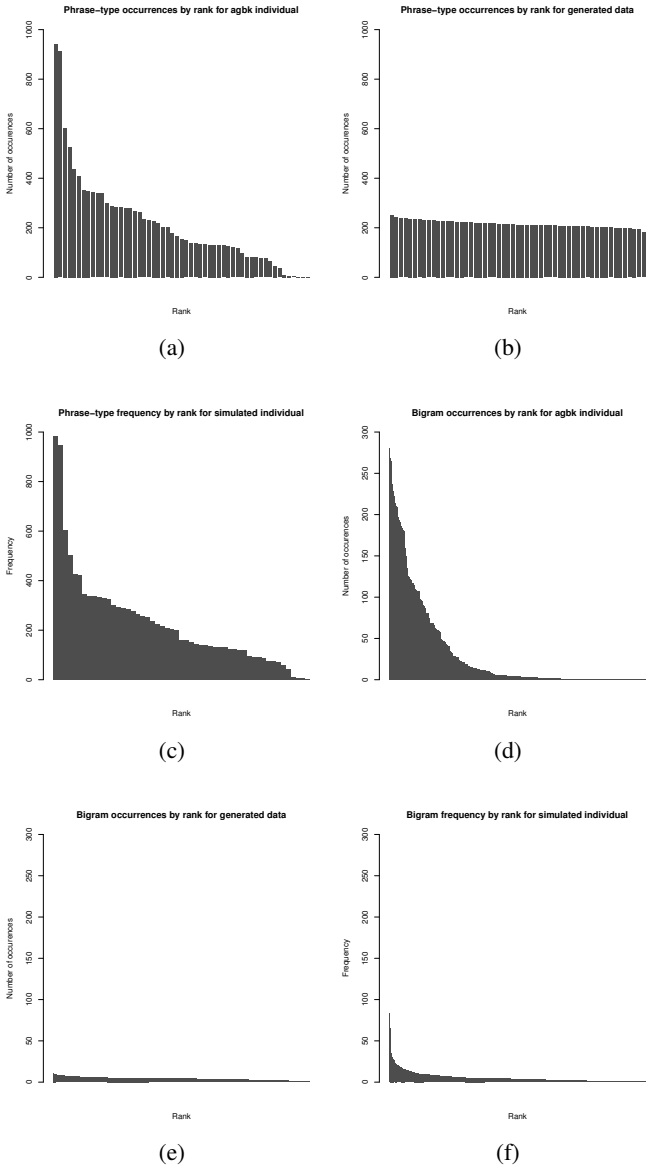

Figure 2: Phrase-type and bigram occurrences sorted by rank out of sequences of 11838 phrases. (a) Observed phrase-types in *agbk* samples; (b) Generated phrase-types in negative samples from uniform distribution (S_{-_1}); (c) Generated phrase-types in negative samples from same distribution as observed samples S_{-_2}. (d) Observed bigrams in *agbk* samples; (e) Generated bigrams in negative samples from uniform distribution (S_{-_1}). (f) Generated bigrams in negative samples from same distribution as observed samples (S_{-_2}).

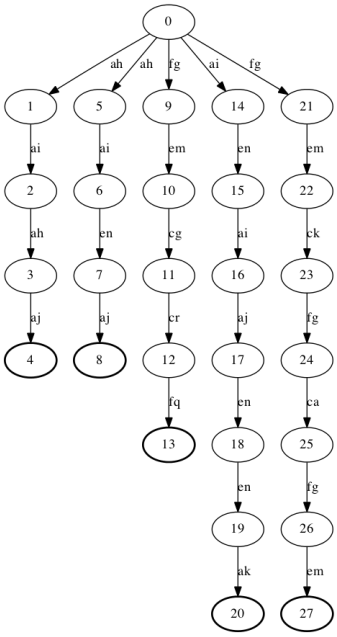

Figure 3: Maximal Canonical Automata derived from sample sequences shown in Table 1. Nodes with a thicker line denote acceptor states.

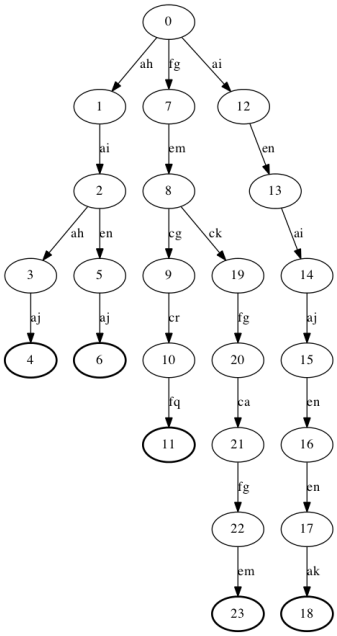

Figure 4: Prefix Tree Acceptor derived from sample sequences shown in Table 1. Nodes with a thicker line denote acceptor states.

- $\bar{q_\lambda}$ is the initial state.
- $\bar{\mathbb{F}}$ is the set of final states.

In short, given a partition $\Pi = \langle P_0, P_1, ..., P_k \rangle$, in which $P_i, i \leq k$ represents a group of states from Q, the automata $a/\bar{\Pi}$ is the result of merging together all of the states in each P_i. Individuals were encoded as a string $\langle x_0, x_1, ..., x_n \rangle$, where x_i represents the *group* to which the corresponding node in the PTA belongs and n is equal to the total number of nodes in the PTA. For example, the partition [[2, 21], [20], [7], [0], [14, 17], [15], [19], [6], [8], [13], [9, 10, 11, 18], [5, 16], [3], [12], [4], [1]] is encoded by the string $\langle 3, 22, 0, 16, 19, 14, 8, 2, 9, 13, 13, 13, 17, 12, 4, 6, 14, 4, 13, 7, 1, 0 \rangle$. See Figure 5.

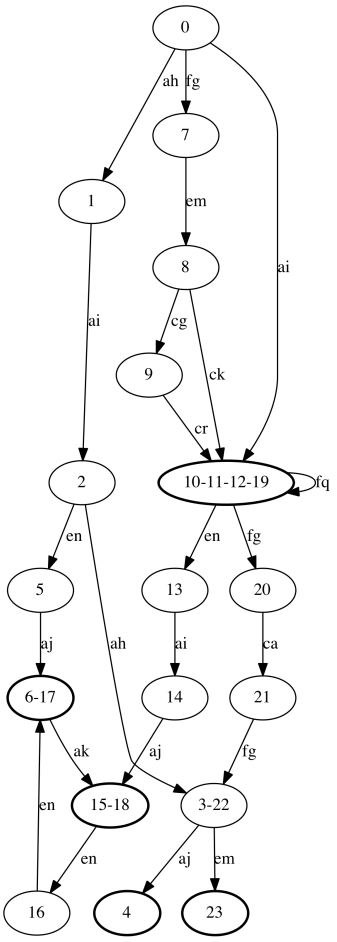

Figure 5: Automata derived from applying partition $\langle 3, 22, 0, 16, 19, 14, 8, 2, 9, 13, 13, 13, 17, 12, 4, 6, 14, 4, 13, 7, 1, 0 \rangle$ to the automata shown in Figure 4. Nodes with a thicker line denote acceptor states.

Thus, the initial population was generated as a collection of 100 random partitions over the PTA.

Genetic Operators

Crossover: each generation, pairs of individuals from the population are chosen through tournament selection with a size of three and a probability of 95% and reproduced by swapping their chromosomes at a randomly selected crossover point. Mutation: individuals in the population change the partition group to which one of their nodes belongs with a probability of 1%.

Both of these genetic operators are guaranteed to produce valid individuals since all partitions of the PTA represent a valid solution for the formulated problem.

Fitness Function

Individuals are evaluated by a combined measure of the underlying DFA they represent, given by its complexity and inconsistency. The goal of the GA is to minimise this combined score.

The complexity of a DFA was measured following the Minimum Description Length (MDL) principle. The MDL principle states that the best solution for a given set of data is that which minimises the encoding of the data (Rissanen, 1978); when applied to grammatical inference, this means that the best hypothesis for some observed data S_+ is that which minimises the encoding of the grammar and its parsing of the data (De la Higuera, 2010). Specifically, we defined our complexity function for a DFA A as described in Equations 1 and 2.

$$complexity = |Q| \times |\Sigma| + \sum^{s} d(q_i) \tag{1}$$

$$d(q) = \begin{cases} \log\left(|\{a \in \Sigma : \delta(q,a)\text{is defined}\}|\right) & \text{if } q \in \mathbb{F}_R \\ \log\left(1 + |\{a \in \Sigma : \delta(q,a)\text{is defined}\}|\right) & \text{if } q \in \mathbb{F}_A \end{cases} \tag{2}$$

In Equation 1, $|Q|$ denotes the number of states of A; $|\Sigma|$ denotes the size of the alphabet, and $\sum^s d(q_i)$ the sum of the respective complexities $d(q_i)$ of each state q_i visited while parsing a sample s. Equation 2 represents the number of possible decisions presented in each state q as signified by the number of exiting edges for the state, plus one when the state is an acceptor, as ending parsing (and thus accepting the string s) represents an extra option. Note that states with only one option (i.e. only one exiting edge or an acceptor state with no exiting edges) will have a $d(q) = log(1) = 0$.

The inconsistency of a DFA is measured as the proportion of negative samples it accepts.

$$inconsistency = \frac{|B_{acc}|}{|B_-|} \tag{3}$$

In Equation 3, $|B_{acc}|$ denotes the number of negative samples accepted and $|B_-|$ the total number of negative samples presented.

Two fixed-length random subsamples from S_+ and S_- are thus used each generation to evaluate the complexity and inconsistency respectively of the individuals in the population.

Results

Our approach proved to be successful at continuously decreasing the complexity of the target solution while also lowering the proportion of negative samples wrongly accepted. Figures 6 and 7 (a) and (b) show the box plot distributions of the complexity and inconsistency scores for the possible solutions population at each generation. For both the *gay* and *meadow* individuals a steady decline in inconsistency can be appreciated, however, the decrease in complexity is relatively low from generation to generation, making it necessary to either significantly increase the size of the initial population or the total number of generations the algorithm is allowed to run. Each of these options imply a significant overhead in computational cost due to the required manipulation of automata with a high number of nodes.

To overcome this we instead opted for an alternative iterative approach. Starting with an initial population of size $p = 100$, after $n = 10$ generations the best found solution was used as the starting point for generating a new population of a subsequent run, or *round*, of the GA, with an initial population of $p = p + 50$. After $n = n + 5$ generations, the best found solution is once again used as the starting point for another round of the GA. These values were determined both empirically and to match the available computational resources.

Figures 6 and 7 (c) and (d) show the box plot distributions of the complexity and inconsistency scores for each generation of the second round. Both scores follow a similar tendency as the previous round, with complexity scores improving slightly at each generation, while inconsistency scores show a speedier improvement. Figures 6 and 7 (e) and (f) show the results for a third round. Although the overall best possible solutions belonged to populations in this round, several individuals in those populations suffered over generalisation problems, as demonstrated by the increased ratio of incorrectly accepted negative samples and significantly low complexities.

A drawback to this approach is that it limits the search space on each iteration, from the lattice over the PTA to the lattices over the partial solutions, in doing so, better solutions might become completely inaccessible to the search procedure. However, it also offers the advantage of arriving at better solutions faster, expending less computational resources.

Figures 8 (a) and (b) show how the most significant decreases in the number of states for the best found solution were a direct result of the start of a new round (denoted by the dotted lines). Likewise, Figures 8 (c) and (d) show the same behaviour for complexity scores. The same cannot be said for inconsistency scores shown in Figures 8 (e) and (f),

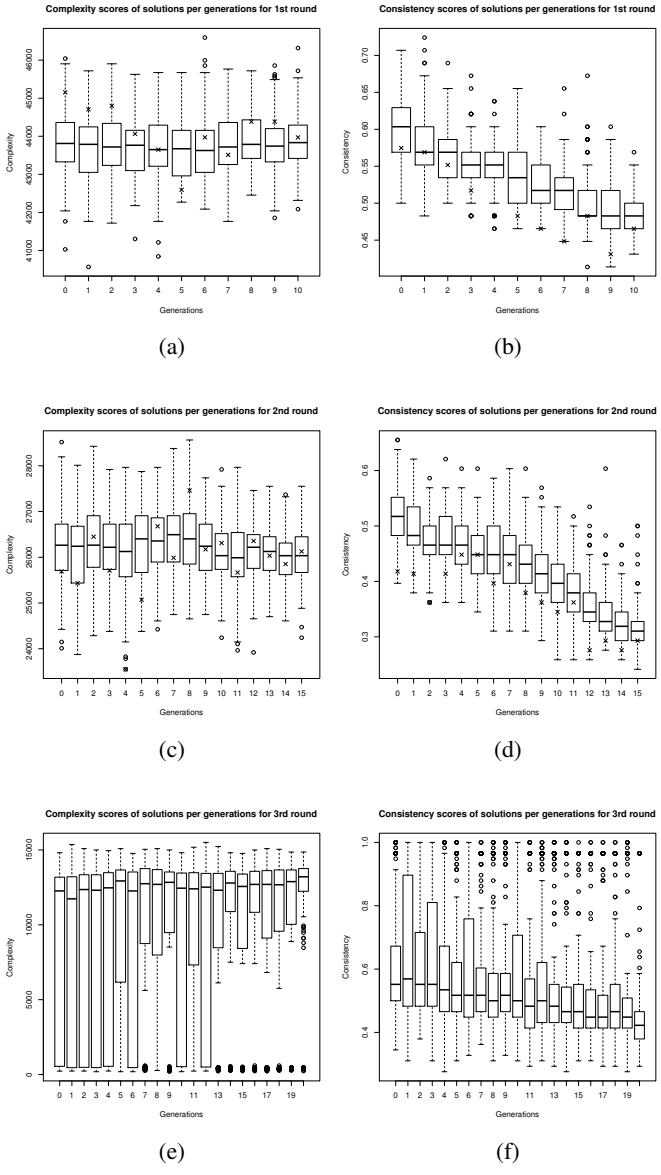

Figure 6: Box plot distribution of complexity and inconsistency scores across all individuals in population throughout each generation after one (a and b), two (c and d), and three (e and f) rounds *ayo* individual using negative sample S_{-_1}. The best solution found for each generation is marked with an x.

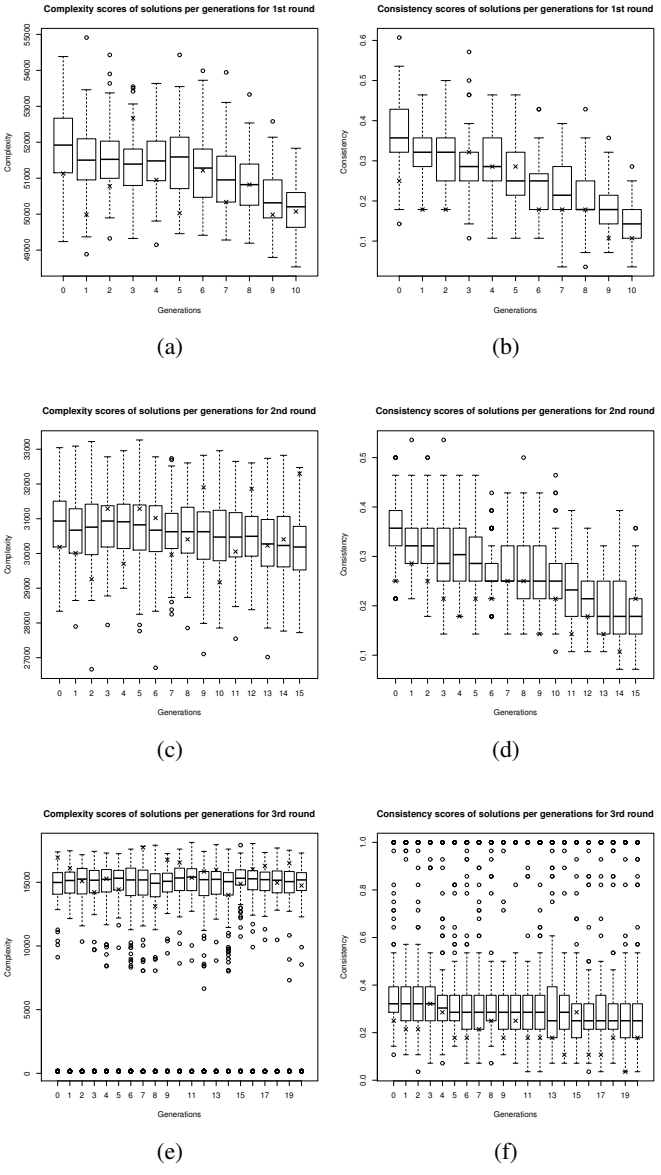

Figure 7: Box plot distribution of complexity and inconsistency scores across all individuals in population throughout each generation after one (a and b), two (c and d), and three (e and f) rounds *meadow* individual using negative sample S_{-_2}. The best solution found for each generation is marked with an x.

where rounds one and two follow a more continuous decline until round three, where the over generalization of the potential solutions causes a sudden increase.

Discussion and Future Work

In this work, we explored the capabilities of a genetic algorithm coupled with the minimum description length principle to model an approximate representation of the syntactic structure governing *CaVi's* songs. Moreover, we analysed the results in terms of complexity and inconsistency of the evolved models.

Regarding the experiments comprising random negative examples, by sampling both phrases and their position in the sequence from uniform distributions, we observed a considerable reduction in complexity as a results of the evolutionary process. The inconsistency of the model also reduced considerably, often maintaining this tendency even with models of reduced complexity.

However, the syntactic models that were evolved using negative examples where phrases were sampled from the phrase distribution of the positive examples at random positions, began suffering from overgeneralisation as the complexity decayed.

The choice of using a DFA as a target representation was based on its simplicity, as there is no definite knowledge of the underlying complexity of the CaVi's syntax. The inability of DFAs to improve consistency when there is some degree of similarity between the positive and negative examples, suggests that more complex models such as probabilistic DFAs, pushdown automata or linear bounded automata, should be considered in the future. Similarly, this method uses exclusively a genetic algorithm for the search of possible solutions, however other machine learning techniques, such as tabu search, could be used instead. In the future we will experiment with different combinations of target representations and search mechanisms and compare their strengths and weaknesses among themselves and other existing methods.

Several obstacles hinder the study of this problem. Data availability remains a major concern, as the process required for gathering and preprocessing is extremely time consuming. This lower availability of data complicates the usage of already established techniques for grammatical inference as the field is more oriented towards human languages, for which bigger data sets are easier to come by.

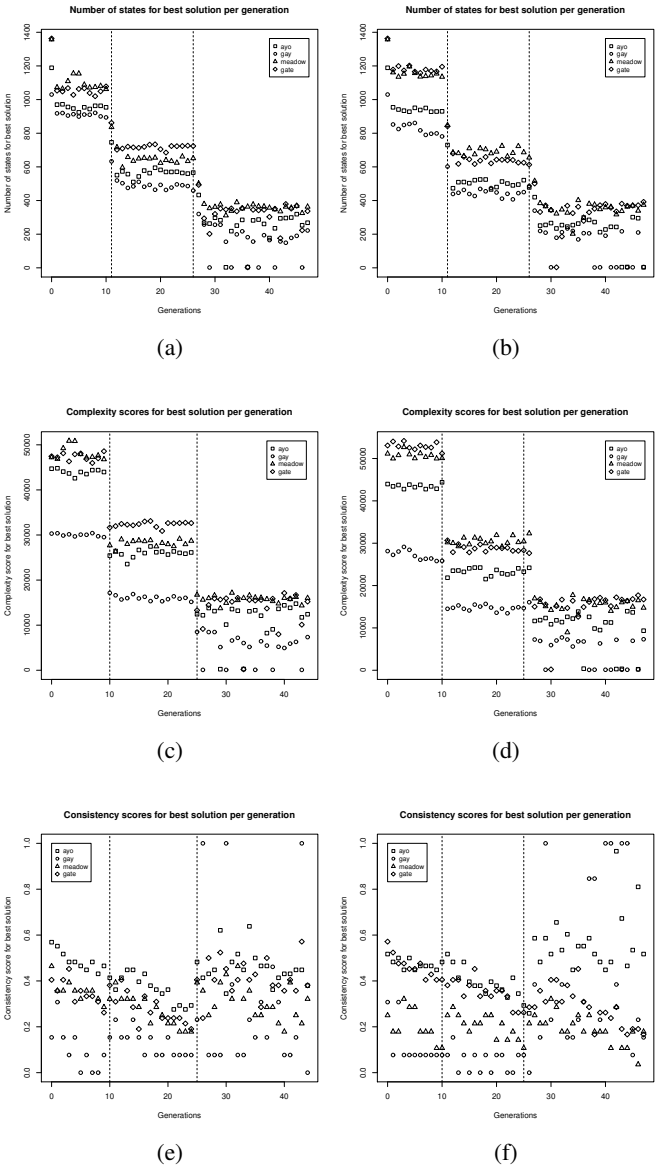

Figure 8: Number of states in best solution per generation for four different individual: *ayo, gay, meadow* and *gate*. Dotted lines along y axis represent different rounds of the GA. (a), (c), and (e) used negative samples S_{-_1}; (b), (d), and (f) used negative samples S_{-_2}.

Acknowledgments

This work was supported by the US National Science Foundation under Award Number 1125423 and by Consejo Nacional de Ciencia y Tecnología under Award Number I010/214/2012.

References

Arriaga, J. G., Kossan, G., Cody, M. L., Vallejo, E. E., and Taylor, C. E. (2013). Acoustic sensor arrays for understanding bird communication. identifying cassin's vireos using svms and hmms. In *Advances in Artificial Life, ECAL*, volume 12, pages 827–828.

Arriaga, J. G., Sanchez, H., Vallejo, E. E., Hedley, R., and Taylor, C. E. (2016). Identification of cassin? s vireo (vireo cassinii) individuals from their acoustic sequences using an ensemble of learners. *Neurocomputing*, 175:966–979.

Berwick, R. C., Okanoya, K., Beckers, G. J., and Bolhuis, J. J. (2011). Songs to syntax: the linguistics of birdsong. *Trends in cognitive sciences*, 15(3):113–121.

Boersma, P. and Weenink, D. (2014). Praat: doing phonetics by computer. Computer program.

De la Higuera, C. (2010). *Grammatical inference: learning automata and grammars*. Cambridge University Press.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Goguen, C. B. and Curson, D. R. (2002). *Cassin's vireo: Vireo cassinii*. American Ornithologists' Union.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and control*, 37(3):302–320.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA.

Hedley, R. W. (2016). Composition and sequential organization of song repertoires in cassin's vireo (vireo cassinii). *Journal of Ornithology*, 157(1):13–22.

Honda, E. and Okanoya, K. (1999). Acoustical and syntactical comparisons between songs of the white-backed munia (lonchura striata) and its domesticated strain, the bengalese finch (lonchura striata var. domestica). *Zoological Science*, 16(2):319–326.

Kakishita, Y., Sasahara, K., Nishino, T., Takahasi, M., and Okanoya, K. (2009). Ethological data mining: an automata-based approach to extract behavioral units and rules. *Data Mining and Knowledge Discovery*, 18(3):446–471.

Kantapon, K., Tan, L. N., Alwan, A., and Taylor, C. E. (2015). A robust automatic phrase classifier using dynamic time-warping with prominent region identification. *Proceedings of ICASSP*, In press.

Pitt, L. and Warmuth, M. K. (1989). The minimum consistent dfa problem cannot be approximated within and polynomial. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 421–432. ACM.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Sasahara, K. and Ikegami, T. (2003). Coevolution of birdsong grammar without imitation. In *Advances in artificial life*, pages 482–490. Springer.

Sasahara, K. and Ikegami, T. (2004). Song grammars as complex sexual displays. In *Artificial Life IX: Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems*, pages 194–199.

Tan, L. N., Alwan, A., Kossan, G., Cody, M., and Taylor, C. E. (2015). Dynamic time warping and sparse representation classification for birdsong phrase classification using limited training data. *Journal of the Acoustic Society of America*, In press.

Taylor, C. E. and Cody, M. L. (2015). Bird song: a model complex adaptive system. *Artificial Life and Robotics*, 20(4):285–290.

van Rossum, G. and Drake, F. (2001). Python reference manual. Technical report, PythonLabs, Virgina, USA.