Ensemble Techniques for DFA Learning

DFA Ensembles without Suitability Metrics

Georgios Tsampikos Kontos[1]
Supervisor(s): Sicco Verwer[1]**, Simon Dieck**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 15, 2025

Name of the student: Georgios Tsampikos Kontos
Final project course: CSE3000 Research Project
Thesis committee: Sicco Verwer, Simon Dieck, Merve Gurel

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Deterministic Finite Automata (DFAs) are interpretable classification models, typically learned through merging states of a large tree-like automaton, an Augmented Prefix Tree Acceptor (APTA), according to heuristic suitability metrics. This paper introduces an ensembling approach for DFAs that does not depend on such heuristics. Starting from the APTA, we construct diverse automata by applying randomized sequences of state merges, while avoiding repetition of merges whenever possible. We also propose a novel graph-connectivity-based metric for inter-model variety. Experimental results on the STAMINA competition datasets yield improved predictive performance compared to models learned using state-of-the-art heuristics on sparse datasets, as well as a tight connection between inter-model variety and performance.

1 Introduction

Deterministic Finite Automata (DFAs) and their variants are interpretable models that can be utilized for classification and prediction purposes. Originating from the theory of computation [1], DFAs have several applications [2] in engineering advancements. For example, finite automata have previously been used for cybersecurity protocol analysis in the case of HTTP [3] and in lexical representation [4] for natural language processing.

The standard passive learning approach to DFA learning [5] involves building a tree-structured automaton from the data, and then merging suitable states to simplify the automaton. Heuristic rules determine the best merges to apply. The rules could either be specific to the domain of state merging, like Evidence-Driven State Merging (EDSM) [6], or well-established statistical rules like the Akaike Information Criterion (AIC) [7]. While heuristics have proved very effective, they are also prone to overfitting on the training data.

Given the broad applicability of DFAs, improving their performance and generalization is of great interest from a scientific standpoint. This motivates the exploration of ensemble learning approaches, which have been more successful than traditional models in dealing with imbalanced and noisy data [8]. In this paper, we investigate the construction of ensembles of DFAs that do not employ any suitability heuristic. Instead of creating a single automaton by selecting the "best" merge at every step, we aim to construct multiple diverse automata by randomly merging states. A similar algorithm has been developed for another type of interpretable model, decision trees, in the form of Extremely Randomized Trees [9].

We construct ensembles of DFAs by applying different merges on the automata from the early stages and exploring as many diverse sequences of merges as possible. We also examine the effect of imposing a uniqueness constraint on the merges applied to each automaton, where the same two states are not allowed to be merged in the construction of several DFAs. Furthermore, we introduce a new metric for inter-model variety by constructing a weighted graph from prediction outputs and evaluating the strength of connections between its vertices. The predictive performance and model diversity of the proposed algorithms are evaluated on datasets of various sparsity levels.

Section 2 introduces the main concepts of DFA learning and the problem formulation. Our novel ensemble algorithm is presented in section 3. The conducted experiments and their results are displayed in section 4. Section 5 discusses the experiment outcomes. Section 6 concludes the contribution of this paper.

2 Background and Problem Formulation

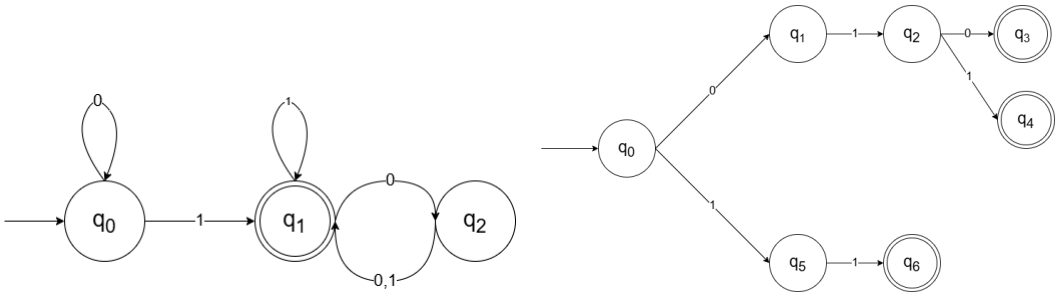
This section provides the necessary background for understanding the problem tackled in this paper. Subsection 2.1 formally defines DFAs and the standard approach to learning a DFA from data. Finally, subsection 2.2 presents the issue of a random ensemble for DFA learning.

2.1 Learning Deterministic Finite Automata

A DFA is a machine that, given a string of input symbols, processes the string one symbol at a time by transitioning within a finite set of states, and determines whether to accept the string by the final state. More formally:

Definition 1. *[10] A* Deterministic Finite Automata (DFA) *is a 5-tuple* $(Q, \Sigma, \delta, q_0, F)$*, where:*

- Q *is the finite set of* states*,*
- Σ *is the finite set of symbols the automation understands, called the* alphabet*,*
- $\delta : Q \times \Sigma \to Q$ *is the* transition function*,*
- $q_0 \in Q$ *is the initial state,*
- $F \subseteq Q$ *is the set of* accept states*.*



(a) 3-state DFA (b) Prefix Tree

Figure 1: (a) A simple 3-state DFA. In this example, $Q = \{q_0, q_1, q_2\}$ and $\Sigma = \{0, 1\}$. The starting state, q_0, is noted by a plain incoming arrow. The set of accepting states is the singleton $F = \{q_1\}$, and a double circle denotes every accepting state. The transition function, δ, is given by the arrows and their labels. (b) An example of an augmented prefix tree acceptor. A set of positive samples that would generate this APTA is $\{011, 010, 11\}$. In this APTA, states q_4 and q_6 could be merged.

The common (passive) approach to learning a DFA from data begins with building a tree-shaped automaton. An example of such a DFA is shown in Figure 1b. This tree-shaped DFA can be defined as follows:

Definition 2. *[11] An* Augmented Prefix Tree Acceptor (APTA) *is a DFA such that the computations of two strings, s and s', reach the same state q if and only if the two strings share the same prefix until they reach q.*

APTAs are considered augmented because they may contain states that are yet unknown whether they are accepting or rejecting. After the APTA is constructed, its states are merged iteratively until no merge is possible, to identify a smaller DFA. Following the red-blue framework, every state of the APTA is colored either red or blue. There are two types of operations: merging a red state with a blue state and upgrading a blue state to a red state. Two states, p and q, can only be combined into one if they are *consistent*; if p is accepting and q is rejecting, or vice versa, the merge is not possible. Different heuristics, including Evidence-Driven State Merging (EDSM) [6] and Alergia [12], can be used to determine which operations are most suitable. For the remainder of this paper, we refer to both types of operations (merging two states or changing the color of a state) as *merges*.

2.2 Problem Statement

In this paper, we propose an ensemble approach to DFA learning that operates without relying on suitability heuristics. After constructing the APTA $\mathcal{A}$ from the (training) data, we generate multiple DFAs by applying different sequences of merges to $\mathcal{A}$ until no more merges are possible. Because no heuristic is used to guide the merge selection, all valid merges are considered equally reasonable.

While the individual DFAs produced by this ensemble method are expected to underperform in comparison to automata learned using state-of-the-art heuristics, the ensemble may benefit from the increased diversity among its members. The inter-model variety can be leveraged through prediction aggregation methods, such as the majority vote.

The premise of our approach is that diversity among models can lead to better generalization, particularly in scenarios where the training data is sparse. In such settings, models guided by heuristics are prone to overfitting and can result in poor predictive performance. By contrast, an ensemble of DFAs can capture a broader range of patterns, thereby reducing systematic errors and enhancing robustness.

3 Ensembling by Randomized State Merging

The contributions of this work are presented in this section. Subsection 3.1 explains the trivial approach of creating randomized DFAs independently. An auxiliary data structure used for controlling merge sequences is defined in subsection 3.2. In subsection 3.3, an ensemble algorithm that balances how many DFAs repeat the same merges is introduced. Subsection 3.4 demonstrates how this approach can be modified to avoid merging the same states in different ensembles.

3.1 Independent Merge Sampling

The most straightforward approach to constructing an ensemble for DFAs is to perform thoroughly and independently random sequences of merges. Starting from the initial APTA $\mathcal{A}$, at each step a merge is drawn uniformly at random and applied, until there are no more possible merges. This procedure is repeated independently to produce the desired number of DFAs.

Compared to more involved approaches, this method is efficient and straightforward. It places no assumptions on which merges are likely to be beneficial. However, the merge sequences for each DFA are chosen independently, meaning that inter-model variety is not

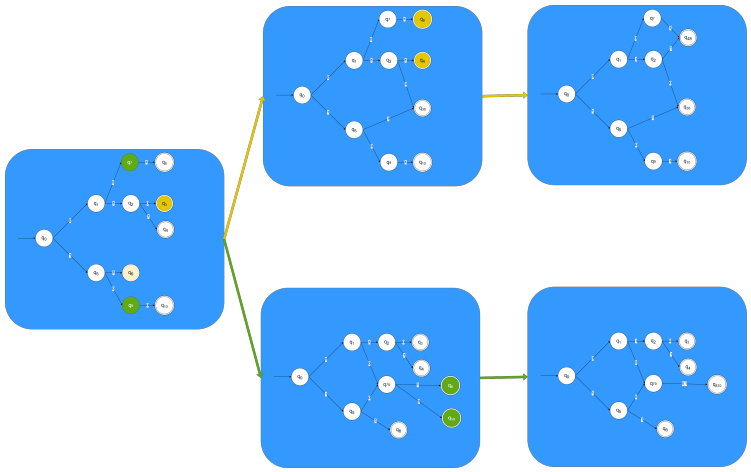


Figure 2: Visualization of independent merge sampling for creating two DFAs. The original APTA $\mathcal{A}$ is depicted in the leftmost DFA. The merges selected for the first DFA are shown in yellow, and the merges selected for the second DFA are shown in green.

encouraged in any way. As such, this algorithm serves as a reference point for assessing the impact of inter-model variety on the ensemble's performance.

3.2 Merge Trees: A Data Structure for Managing Merge Sequences

While iterative DFA construction is well-suited for independent merge sampling, it poses significant challenges when attempting to enhance the ensemble's inter-model variety. Designing each new DFA to be as diverse as possible from those already in the ensemble is an inherently difficult task. In intermediate steps, automata still contain redundant states, making comparisons with entirely constructed DFAs skewed and uninformative. The alternative approach of constructing entire terminal DFAs at each iteration and evaluating their similarity with the models already in the ensemble would require executing complete merge sequences repeatedly, resulting in very high computational complexity.

To design more structured ensemble methods that aim to maximize the inter-model variety, a systematic way to track and organize merge selections is required. We introduce the merge tree, a tree-like data structure that compactly represents merge sequences from the original APTA and remembers which DFAs of the ensemble have applied each merge. Merge trees are formally described by the following definition:

Definition 3. *A* merge tree T *is a rooted tree that encodes sequences of state merges performed to construct finite automata from an APTA* $\mathcal{A}$*. Each node in* T *corresponds to a partial sequence of merges, representing an (intermediate) automaton. Each edge from a parent to a child is labeled by the merge operation that transforms the parent's DFA into the child's DFA. The root corresponds to the initial automaton* $\mathcal{A}$*. The children of each node correspond to the DFAs obtainable by a single feasible merge from the DFA associated with the parent node.*

Maintaining a merge tree enables efficient and incremental exploration of merge sequences. Prefixes of merge sequences can be reused to reduce redundancy. Rather than constructing

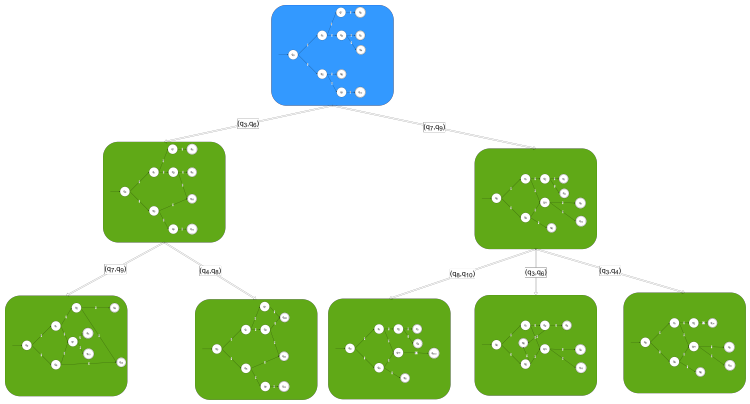


Figure 3: (Partial) merge tree. The levels of the tree are presented left to right, with the leftmost node being the root. Here, the original APTA is the same as that of Figure 2. Every edge of the tree is labeled with the corresponding pair of states that should be merged to transition to the child state from its parent state.

DFAs sequentially, we can build multiple automata in parallel by making shared intermediate decisions. A thorough exploration of the tree ensures that only essential merges are performed, thereby avoiding unnecessary computation. The task of constructing an ensemble with high inter-model variety reduces to the problem of selecting diverse leaves of the merge tree.

3.3 Balanced Merge Tree Exploration

Using the previously described merge tree data structure, we can construct an ensemble of n automata by exploring different branches in a balanced manner. The core idea of the *Balanced Merge Tree Exploration (BMTE)* algorithm is to encourage structural diversity by selecting different merge paths for different automata.

We build all the automata of the ensemble by performing a level exploration of the merge tree. The tree is explored depth-first, so that reverting merges, which for deep trees increase the computational complexity, are avoided whenever possible. Every visited node maintains a set of DFAs that have followed the corresponding sequence of merges (the path from the root to the node). We say that those automata are *live* on the node. Naturally, all n automata are live on the root. We call visited nodes that are assigned at least one DFA *selected*, and visited nodes that are not assigned any automata *skipped*.

The algorithm maintains a stack of selected nodes and processes them sequentially. Initially, the stack only contains the root. If the node in hand is a leaf, it is added to the ensemble. If it is an inner node, the algorithm allocates all the live DFAs of the node along its children. The allocation is randomized, but it seeks to distribute the live DFAs evenly across the children, covering as many branches of the merge tree as possible. An example of such an allocation is shown in Figure 4. Once the allocation is complete, all selected children are added to the stack. This algorithm continues until n DFAs are added to the ensemble.

In theory, an unfortunate allocation may lead to live DFAs arriving at the same leaf. In that case, only one of those DFAs would be added to the ensemble, and hence, the desired number of automata will not be reached. We can bypass this process by retrieving the nodes

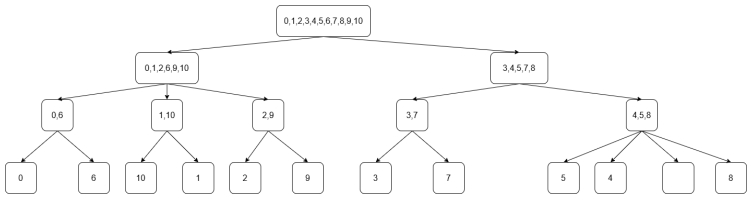


Figure 4: An example of a balanced allocation. The 11 DFAs that are live in the root are distributed across its two children, so that the left child receives 6 DFAs and the right child receives 5. The same logic is followed for the rest of the nodes. The rightmost node of the third level has three live DFAs (4, 5, and 8) and four children. As a result, one of the children (third from left to right) is skipped.

of the merge tree that were skipped during allocation and performing a random walk until we reach the leaves. In practice, this situation is improbable.

If the original APTA has m states, then the number of possible state merges is bounded by $\mathcal{O}(m^2)$, as there exist $m(m-1)$ pairs of distinct states. This implies that to reach a terminal DFA from the APTA, at most $\mathcal{O}(m^2)$ must be performed. If a merge can be performed and reverted in $\mathcal{O}(k)$ time, then the theoretical time complexity of the BMTE algorithm is $\mathcal{O}(nm^2k)$. In practice, backtracking branches of the merge tree allows us to reduce the number of merges we apply and revert.

3.4 Repeated Merges and Branch Pruning

The BMTE approach is designed to cover a wide variety of merge sequences by encouraging divergence among the DFAs in merge sequences from the early stages of the merge process. However, a key limitation is that identical merges can still occur in different levels of branches, leading to structural similarities between the DFAs. This can, in some cases, reduce the inter-model variety, as structural similarities may be related to language similarities [13].

For example, let node c have three live DFAs, $0, 1, 2$, and two feasible merges, a and b, resulting in child nodes c_a and c_b respectively. The algorithm might assign DFA 0 to c_a, and DFAs 1 and 2 to c_b. If merge a is also feasible at c_b, one of the DFAs from c_b might be allocated to the corresponding child, c_{ba}. Although the DFAs corresponding to c_a and c_{ba} follow different merge sequences and hence are not identical, they may exhibit structural overlap, since they both include the merge a in their merge sequence. This undermines the inter-model variety of the ensemble by increasing the likelihood of similar automata being constructed.

To address this issue, we maintain a global memory of previously performed merges. Whenever a DFA is added to the ensemble, we add all the merges in its merge sequence to the memory. Then, before allocating the live selections of a selected node to its children, we examine the merges that lead to those children. If a child's defining merge is already present in the memory, we *prune* the child, eliminating it from the allocation process.

In the latter stages of the algorithm, when the memory becomes large, it is possible that all of the children of a selected node are pruned. In that case, a dead end occurs; we cannot allocate the live DFAs to the children without repeating some merges. Instead, we maintain a set of DFAs that reach dead ends, and whenever a child of a selected node is skipped, we

assign it one of these DFAs. The process is detailed in Algorithm 1. The theoretical time complexity has the same asymptotic bound as BMTE, namely $\mathcal{O}(nm^2k)$.

Algorithm 1: Balanced Merge Tree Exploration with Branch Pruning

Input: An APTA $\mathcal{A}$ and number of estimators n
Output: A list $\mathcal{E}$ of n DFAs

Initialize: $\mathcal{E} \leftarrow [\,]$, `memory` $\leftarrow \emptyset$, `afloat` $\leftarrow [\,]$;
Create `root` node with n DFAs; push it onto stack `next_nodes`;
Set `is_reset` $\leftarrow$ `FALSE`, `prev_node` $\leftarrow$ `None`;
while *`next_nodes` not empty* ***and*** $|\mathcal{E}| < n$ **do**
    `node` $\leftarrow$ pop from `next_nodes`;
    if *`is_reset`* **then**
        Undo merges of `prev_node` on $\mathcal{A}$; apply merges of `node`;
    else
        Apply `node.merge` to $\mathcal{A}$;
    Initialize children of `node`; prune those with merges in `memory`;
    if *`node` is a leaf* **then**
        Add merges in `node.sequence` to `memory`; add DFA to $\mathcal{E}$;
    else
        if *all children pruned* **then**
            Add DFAs in `node.live` to `afloat`;
        else
            Allocate live DFAs to children; push selected children to `next_nodes`;
            if *`afloat` not empty* **then**
                Allocate afloat DFAs to skipped children; push them to `next_nodes`;
    Set `is_reset` $\leftarrow$ `TRUE`; `prev_node` $\leftarrow$ `node`;
return $\mathcal{E}$

4 Experimental Setup and Results

This section documents the experiments conducted to assess the effectiveness of the proposed methods. Subsection 4.1 details the experimental setup, including the data and computational environment used to run the experiments. The metrics used to quantify the performance and model diversity of the proposed algorithm are described in subsection 4.2. The results are presented and analyzed in subsection 4.3.

4.1 Experimental Setup

To evaluate the effectiveness and performance of the balanced exploration algorithm proposed in subsection 3.3 and its pruning variant proposed in subsection 3.4, we compare them against two baseline methods. The first is an isolated DFA constructed by EDSM [6], a state-of-the-art heuristic for state merging, which uses positive and negative samples to examine state equivalence by separating states into two groups (blue and red states). The second approach is the random walk method for DFA ensembling, described in subsection

3.1, which ensembles DFAs without coordination. For all methods, the ensemble size is set to 100 estimators. Given the inherited randomness of the ensemble algorithms, we repeated each experiment 10 times and averaged the results.

Our experiments are conducted on 20 independent datasets derived from the STAMINA competition [14]. These datasets are generated by state machines that represent software models. The varying sparsity (ratio of positive samples), alphabet, and sample size of these datasets make them a reliable benchmark suite for DFA inference. We split each set into a training and test with an 80/20 proportion, maintaining the sparsity of the original dataset. The training sets are used to construct the DFAs and ensembles, while the test sets are reserved exclusively for evaluation.

All experiments are conducted using FlexFringe [15], a C++ framework for learning finite state machines from data. FlexFringe includes a built-in implementation of the EDSM algorithm, which can be used directly. Both ensemble approaches are implemented as extensions to FlexFringe as part of this paper.

The experiments were conducted in the Delft Blue supercomputer [16], utilizing 12 computing cores with 3 GB of memory allocated to each core.

4.2 Evaluation Metrics

We evaluate the ensemble methods in terms of both predictive performance and inter-model diversity. In other words, we are not only interested in how accurate the aggregate predictions of an ensemble are, but also in how much the predictions of each automaton in the ensemble differ. In this subsection, we summarize the metrics used to quantify these attributes.

In the context of recognizing the outputs of an automaton, positive and negative samples do not have specific meanings. We can measure how effective an algorithm is in capturing true positive and true negative samples by the *sensitivity* and *specificity* metrics:

$$\text{Sensitivity} = \frac{TP}{TP + FN}, \text{Specificity} = \frac{TN}{TN + FP} \tag{1}$$

The average of the sensitivity and specificity gives us the *balanced accuracy*, which is a performance metric suitable for imbalanced classification tasks:

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{2}$$

Inter-model variety measures are not as well-established as predictive performance metrics. One way to quantify the similarity between two automata is by calculating how often they wrongly agree. The *double fault* ratio captures the common mistakes between a pair of automata. For two DFAs A and B, the double fault ratio is defined as:

$$\text{Double Fault}_{A,B} = \frac{|\text{Misclassified}_A \cap \text{Misclassified}_B|}{N} \tag{3}$$

where N is the sample size, a high double fault ratio indicates not only that the two DFAs are correlated, but that they also harm the overall performance of the ensemble.

We can measure how much the models in the ensemble agree on a single sample by calculating the *entropy* for that sample. For a data point x, the entropy can be calculated as:

$$\text{Entropy}(x) = -p_{\text{accept}}(x) \log p_{\text{accept}}(x) - p_{\text{reject}}(x) \log p_{\text{reject}}(x) \tag{4}$$

where $p_{\text{accept}}(x), p_{\text{reject}}(x)$ are the ratios of models in the ensemble that accept or reject the sample. High entropy indicates greater disagreement on the given point.

Neither of the above metrics provides a comprehensive view of the entire ensemble's diversity across a whole dataset. To this end, we introduce a global measure of the ensemble's coherence, which we define as follows:

Definition 4. *Given an ensemble $\mathcal{E}$, construct a weighted graph $\mathcal{G} = (V, E, w)$, in which each vertex represents a model of the ensemble and each edge is weighted by the fraction of predictions on which the connected models agree. The* agreement connectivity *of the ensemble is defined as the algebraic connectivity[17] of $\mathcal{G}$. We use the notation $\alpha(\mathcal{E})$.*

The algebraic connectivity of a graph $\mathcal{G}$ reflects how connected the graph is. It can be calculated as the second smallest eigenvalue of the Laplacian matrix of $\mathcal{G}$, $L_{\mathcal{G}}$, where the Laplacian is the difference between the weighted diagonal degree matrix, $D_{\mathcal{G}}$ and the weighted adjacency matrix, $A_{\mathcal{G}}$: $L_{\mathcal{G}} = D_{\mathcal{G}} - A_{\mathcal{G}}$. The agreement connectivity for an ensemble $\mathcal{E}$ of n models satisfies $0 \leq \alpha(\mathcal{E}) \leq n$, based on lower and upper bounds of the algebraic connectivity of a connected graph [18].

Algebraic connectivity has previously been used in network analysis [19] to determine how difficult it is for the network to be broken down into independent components. Similarly, agreement connectivity measures how difficult it is to partition the ensemble into groups of models that behave differently. If an ensemble has high agreement connectivity, all its models agree with each other to a large extent. On the other hand, models in an ensemble with low agreement connectivity often deviate from the consensus.

4.3 Experimental Results

The construction of ensembles requires significantly more computation than building a singleton DFA using EDSM. Ensemble construction by random walk and balanced merge tree exploration requires approximately the same computational resources as the complexity of allocating the live DFAs across the children is negligible. In practice, the introduction of pruning results in a significant improvement in runtime, despite the two algorithms having the same asymptotic bounds, as the latter DFAs are constructed very quickly due to the limited number of merge options. The standard deviation of the balanced accuracy across the various iterations of the experiments barely exceeds 0.06, suggesting robust results.

Figure 5 presents the balanced accuracy scores of the various algorithms for each dataset. Figures 6, 7, and 8 show the inter-model variety scores of the ensembles. The datasets are plotted in increasing density. A complete view of the evaluation metrics is available in Appendix A.

In terms of predictive accuracy (Figure 5, BMTE and random walk ensembles perform the best on the majority of datasets. For inter-model variety, BMTE and random walk also outperform pruning BMTE across all metrics, while achieving near identical results with one another.

5 Discussion

The experimental findings show that ensemble-based approaches to DFA learning achieve comparable performance to heuristically learned DFAs. Ensembles appear to outperform

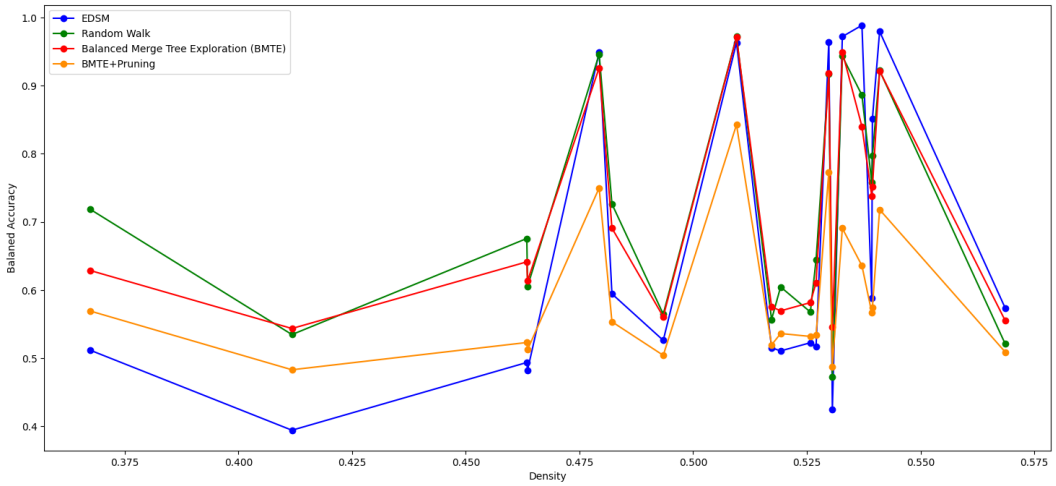


Figure 5: Balanced Accuracy

automata learned by EDSM in imbalanced and sparse datasets, where the evidence-driven strategy of EDSM fails to generalize. In those cases, ensembles excel by aggregating predictions from diverse automata, thereby recovering more true positives, which translates to better balanced accuracy scores, as shown in Figure 5. However, EDSM maintains an edge for denser datasets, approaching near-perfect accuracy in some cases. Overall, ensemble methods generalize better than EDSM and avoid overfitting; still, they are more computationally intense as they examine a vast amount of possible merges.

The results also show that different ensemble methods vary in effectiveness and robustness. The Random Walk ensembles and the BMTE algorithm consistently outperform the BMTE pruning variant. For sparser datasets, the random walk ensemble performs the best on average, while it produces near identical results with BMTE for denser inputs. The balanced allocation of automata across different merges restricts the effect of noisy models on the overall performance by ensuring that only a limited number of DFAs follow the same merges. This results in more robust predictions, as the balanced accuracy scores of BMTE exhibit a lower standard deviation. In contrast, the independent construction of DFAs in the random walk ensemble can result in near-duplicate models, particularly when a small number of feasible merges is available in the early steps. The extreme randomness and independence of DFAs learned by this method also increase the volatility of the algorithm. Furthermore, while BMTE with pruning outperforms the baseline for sparser inputs, it fails to generalize as well as the other two approaches in denser datasets.

A key insight from our findings is that ensemble performance is tightly linked to the inter-model variety. BMTE and random walk ensembles outperform the pruning version of BMTE across all inter-model variety metrics and also consistently achieve better predictive performance. The agreement connectivity and double fault metrics, particularly for BMTE with branch pruning, indicate that ensembles with strongly agreeing automata make many of the same mistakes, leading to lower generalization and worse predictions. The ensembles that perform better are those that include diverse automata, which all capture different patterns from the training inputs.

Even though the pruning augmentation of BMTE appears to enhance diversity by eliminat-

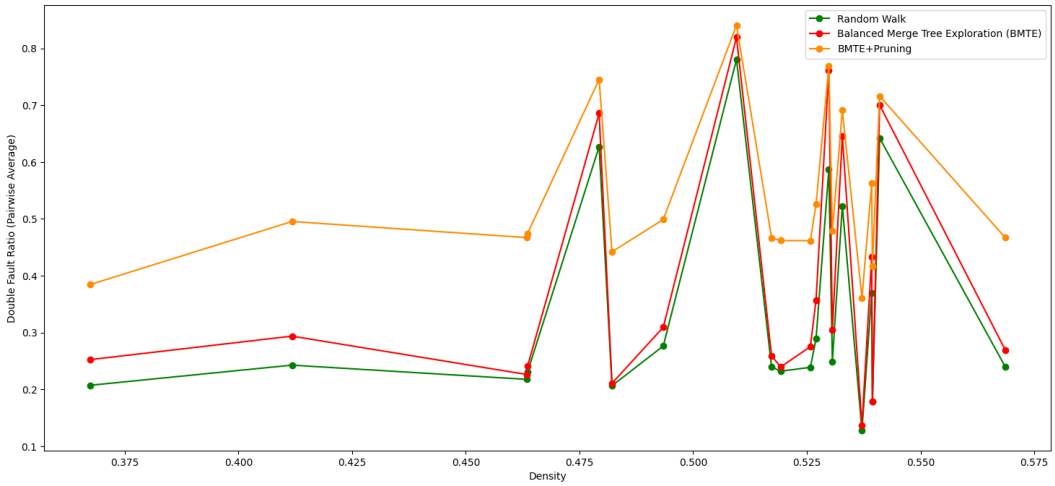


Figure 6: Double Fault Ratio

ing repeated merges and pushing the ensemble to explore distinct refinements to the original automaton, our results reveal the opposite: pruning harms the diversity of the prediction, especially in terms of entropy. This counterintuitive observation demonstrates the distinction between structural and behavioral diversity. Pruning might avoid identical merges, but it does not guarantee a difference in behavior on unseen data. Additionally, the aggressive pruning mechanism may prematurely cut off the high variance of the merge tree. The recurring appearance of merges at different levels of the merge tree could also suggest their increased impact on the DFA's performance. As a result, pruning constructs ensembles that not only tend to agree but also fail together quite often.

6 Conclusions and Future Work

This paper presents an ensembling technique for DFA learning that does not rely on merge suitability heuristics. Starting from the Augmented Prefix Tree Acceptor (APTA) originally learned from the data, our algorithm organizes the space of automata reachable from the APTA in a merge tree, a hierarchical structure where every transition corresponds to a merge of states. The different sequences of state merges that result in the other models in the ensemble are selected such that as many branches of the tree as possible are explored. A modification of the algorithm, which prevents the same merge being performed in different stages by pruning branches of the merge tree, is also proposed.

Moreover, we also introduce a new global metric for the inter-model variety of ensemble classifiers, agreement connectivity, which constructs a graph that details how strongly every pair of models in the ensemble agree. We define the agreement connectivity of an ensemble as the algebraic connectivity of the corresponding graph, which measures how strongly connected its vertices are, and hence, how often the ensemble is close to a consensus. The lower the value of the agreement connectivity, which is always non-negative, the greater the diversity across the automata of the ensemble.

Our experimental studies suggest that the proposed algorithm outperforms the state-of-

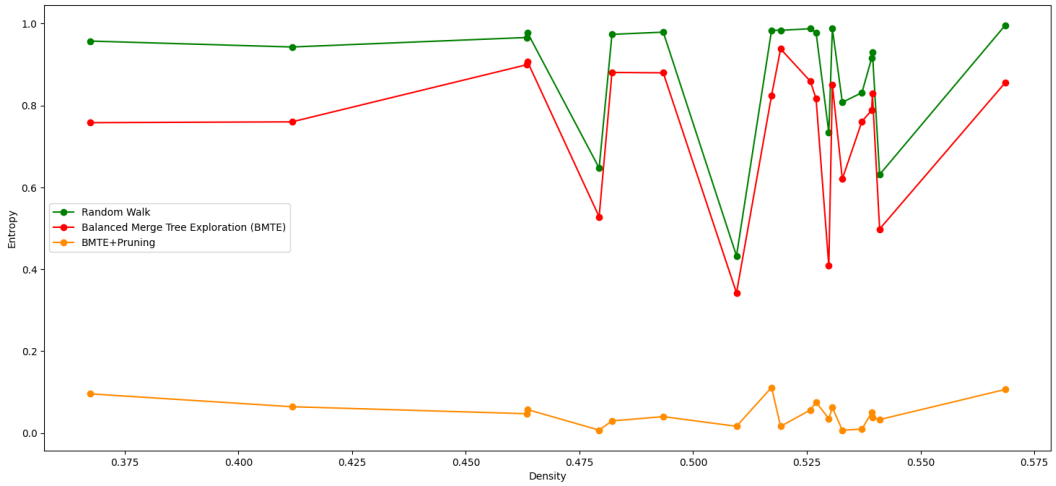


Figure 7: Prediction Entropy

the-art suitability heuristics in sparse inputs, while achieving comparable performance in terms of balanced accuracy in denser datasets. These results support that random walk and balanced merge tree exploration ensembles generalize better than individual heuristically learned models, which are more prone to overfitting on biased training data, and models that prune branches of the merge tree, which hinder inter-model variety.

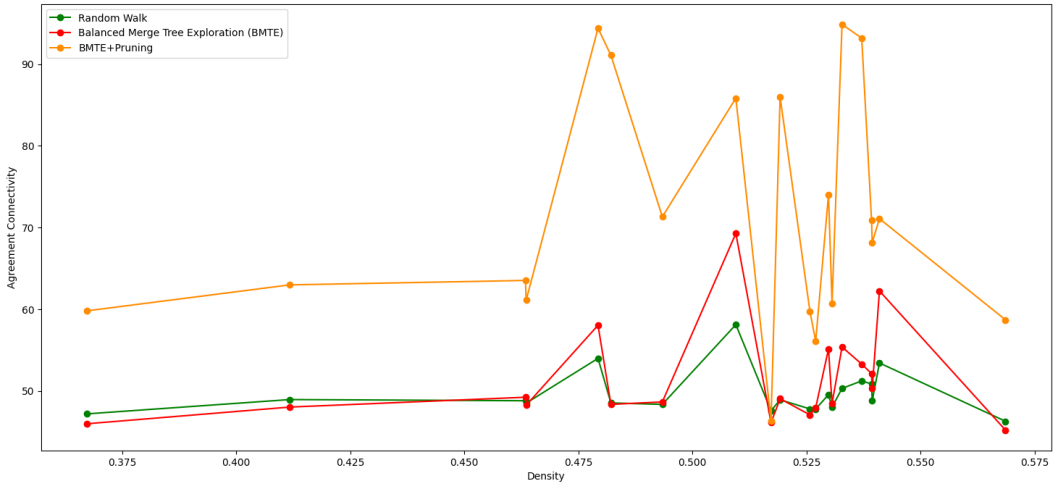


Figure 8: Agreement Connectivity

In the future, our ensemble algorithm may apply to probabilistic variants of deterministic automata (PDFAs), which are theoretically complex but have more real-world applications. The balanced tree exploration can combine with suitability heuristics to balance exploration and exploitation in the merge tree. Additionally, the agreement connectivity metric could extend to regression problems by modifying the graph's weight function.

Responsible Research

Ethical Considerations

As with all data-driven algorithms, there exists the danger of misusing automata. If the proposed algorithms are used in sensitive domains, such as behavior prediction or inference of personal data, they can lead to unethical and malicious programs for surveillance or manipulation. While machine learning models capture patterns and make predictions based on the analytical representations of their inputs, the danger of misuse lies in the applications to which they are put.

Ensembling automata also hinders the overall interpretability of the aggregated predictions. For a single DFA, it is easy to understand its structure and the reasons it accepts or rejects strings. On the other hand, large ensembles contain many automata with varying structures and diverse predictions. The introduction of multiple models and randomness makes interpretations of the algorithm's outcomes significantly more complicated.

Data Availability and Experiment Reproducibility

The research for this paper has been conducted in adherence with the principles of responsible and reproducible research. The data, code, and results used or generated in this study are structured and shared in a transparent and reusable manner.

The code of this research is built on the FlexFringe framework, which is publicly available on GitHub. This repository also includes the datasets from the STAMINA competition that are used in the experiments of section 4. These datasets are synthetically generated and do not originate from real-world data, eliminating privacy concerns.

The modifications to FlexFringe that incorporate the algorithms proposed in this paper are available here. The replication of this work is ensured by sharing the complete experimental pipeline. Apart from the boilerplate FlexFringe code, this repository includes:

- an implementation of the newly introduced data structures and ensemble algorithms.
- dataset splits into training and test sets.
- a Jupyter notebook used to generate these splits.
- the DFA ensembles generated during the experiments.
- predictions of each ensemble on the corresponding test sets.
- a Jupyter notebook used for performance evaluation.
- scripts for running experiments across the datasets and submitting jobs to DelftBlue.

References

[1] M. Sipser, *Introduction to the Theory of Computation*. International Thomson Publishing, 1st ed., 1996.

[2] E. Gribkoff, "Applications of deterministic finite automata," *UC Davis*, pp. 1–9, 2013.

[3] K. L. Ingham, A. Somayaji, J. Burge, and S. Forrest, "Learning dfa representations of http for protecting web applications," *Computer Networks*, vol. 51, no. 5, pp. 1239–1255, 2007.

[4] M. Gross, "The use of finite automata in the lexical representation of natural language," in *LITP Spring School on Theoretical Computer Science*, pp. 34–50, Springer, 1987.

[5] K. P. Murphy, "Passively learning finite automata," tech. rep., 1996.

[6] 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 *International Colloquium on Grammatical Inference*, pp. 1–12, Springer, 1998.

[7] H. Akaike, "Akaike information criterion," in *International encyclopedia of statistical science*, pp. 25–25, Springer, 2011.

[8] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[9] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[10] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM journal of research and development*, vol. 3, no. 2, pp. 114–125, 1959.

[11] M. J. Heule and S. Verwer, "Exact dfa identification using sat solvers," in *Grammatical Inference: Theoretical Results and Applications: 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings 10*, pp. 66–79, Springer, 2010.

[12] R. C. Carrasco and J. Oncina, "Learning stochastic regular grammars by means of a state merging method," in *International Colloquium on Grammatical Inference*, pp. 139–152, Springer, 1994.

[13] P. Grachev, R. Bezborodov, I. Smetannikov, and A. Filchenkov, "Exploring the relationship between the structural and the actual similarities of automata," in *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, pp. 81–86, 2019.

[14] N. Walkinshaw, B. Lambeau, C. Damas, K. Bogdanov, and P. Dupont, "Stamina: a competition to encourage the development and assessment of software model inference techniques," *Empirical software engineering*, vol. 18, no. 4, pp. 791–824, 2013.

[15] S. Verwer and C. A. Hammerschmidt, "Flexfringe: a passive automaton learning package," in *2017 IEEE international conference on software maintenance and evolution (ICSME)*, pp. 638–642, IEEE, 2017.

[16] Delft High Performance Computing Centre (DHPC), "DelftBlue Supercomputer (Phase 2)." https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[17] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[18] D. Spielman, "Spectral graph theory," *Combinatorial scientific computing*, vol. 18, no. 18, 2012.

[19] A. Jamakovic and P. Van Mieghem, "On the robustness of complex networks by using the algebraic connectivity," in *International conference on research in networking*, pp. 183–194, Springer, 2008.

A Detailed Results

Tables 1, 2, and 3 show statistics for the sensitivity, specificity, and balanced accuracy of the different algorithms, respectively. Tables 4, 5, and 6 show statistics for the inter-model variety metrics of the ensembles. EDSM-learned DFAs are excluded from these tables, as they are stand-alone models, not ensembles.

Table 1: Sensitivity

Dataset	EDSM	Random Walk		BMTE		BMTE + Pruning	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
1	0.940	0.926	0.017	0.906	0.030	0.740	0.064
2	0.973	0.898	0.007	0.897	0.013	0.739	0.070
3	0.965	0.986	0.003	0.981	0.005	0.841	0.078
4	0.977	0.932	0.014	0.950	0.024	0.710	0.059
5	1.000	0.890	0.013	0.853	0.018	0.664	0.065
6	0.529	0.628	0.024	0.591	0.011	0.556	0.031
7	0.457	0.712	0.042	0.649	0.054	0.490	0.036
8	0.584	0.734	0.039	0.691	0.047	0.535	0.046
9	0.964	0.888	0.008	0.906	0.030	0.784	0.136
10	0.846	0.771	0.025	0.746	0.051	0.609	0.049
11	0.619	0.723	0.016	0.714	0.055	0.598	0.021
12	0.303	0.467	0.069	0.472	0.056	0.391	0.047
13	0.443	0.631	0.074	0.629	0.037	0.477	0.036
14	0.520	0.561	0.026	0.558	0.023	0.497	0.035
15	0.545	0.586	0.049	0.595	0.022	0.554	0.036
16	0.381	0.749	0.142	0.551	0.093	0.459	0.110
17	0.463	0.512	0.041	0.568	0.026	0.513	0.060
18	0.545	0.645	0.033	0.622	0.027	0.561	0.049
19	0.531	0.572	0.038	0.592	0.084	0.536	0.055
20	0.618	0.587	0.022	0.611	0.019	0.573	0.068

Table 2: Specificity

Dataset	EDSM	Random Walk		BMTE		BMTE + Pruning	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
1	0.957	0.966	0.006	0.946	0.019	0.759	0.058
2	0.985	0.946	0.007	0.946	0.008	0.697	0.083

Continued on next page

Dataset	EDSM	Random Walk		BMTE		BMTE + Pruning	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
3	0.961	0.958	0.005	0.960	0.006	0.844	0.087
4	0.966	0.954	0.013	0.949	0.017	0.671	0.074
5	0.977	0.882	0.016	0.827	0.027	0.607	0.067
6	0.493	0.581	0.025	0.548	0.010	0.516	0.029
7	0.531	0.640	0.017	0.633	0.026	0.556	0.029
8	0.604	0.718	0.032	0.691	0.033	0.572	0.048
9	0.964	0.946	0.010	0.930	0.010	0.763	0.161
10	0.857	0.822	0.022	0.757	0.052	0.540	0.054
11	0.558	0.793	0.035	0.762	0.071	0.537	0.027
12	0.486	0.603	0.019	0.616	0.025	0.575	0.036
13	0.522	0.579	0.020	0.598	0.023	0.548	0.026
14	0.532	0.569	0.021	0.563	0.016	0.511	0.032
15	0.500	0.549	0.057	0.568	0.035	0.510	0.048
16	0.643	0.689	0.013	0.706	0.020	0.680	0.055
17	0.386	0.434	0.072	0.523	0.044	0.462	0.059
18	0.488	0.644	0.055	0.599	0.046	0.507	0.051
19	0.500	0.542	0.041	0.559	0.081	0.503	0.063
20	0.529	0.455	0.038	0.499	0.038	0.445	0.089

Table 3: Balanced Accuracy

Dataset	EDSM	Random Walk		BMTE		BMTE + Pruning	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
1	0.9487	0.9459	0.0085	0.9259	0.0216	0.7498	0.0605
2	0.9794	0.9220	0.0060	0.9215	0.0087	0.7180	0.0763
3	0.9631	0.9721	0.0034	0.9708	0.0038	0.8426	0.0816
4	0.9718	0.9431	0.0122	0.9494	0.0177	0.6906	0.0666
5	0.9885	0.8859	0.0141	0.8399	0.0199	0.6357	0.0660
6	0.5107	0.6046	0.0246	0.5695	0.0086	0.5362	0.0299
7	0.4937	0.6757	0.0281	0.6414	0.0385	0.5231	0.0324
8	0.5944	0.7262	0.0343	0.6908	0.0397	0.5535	0.0470
9	0.9642	0.9170	0.0071	0.9180	0.0178	0.7733	0.1483
10	0.8514	0.7968	0.0211	0.7511	0.0499	0.5747	0.0516
11	0.5886	0.7579	0.0251	0.7383	0.0594	0.5671	0.0236
12	0.3944	0.5347	0.0442	0.5436	0.0400	0.4831	0.0413
13	0.4824	0.6053	0.0462	0.6134	0.0295	0.5124	0.0312
14	0.5262	0.5647	0.0231	0.5609	0.0171	0.5043	0.0333
15	0.5227	0.5678	0.0530	0.5816	0.0274	0.5320	0.0420
16	0.5119	0.7187	0.0754	0.6288	0.0440	0.5694	0.0819
17	0.4247	0.4728	0.0560	0.5456	0.0339	0.4871	0.0593
18	0.5166	0.6442	0.0404	0.6108	0.0348	0.5342	0.0502
19	0.5153	0.5569	0.0392	0.5755	0.0826	0.5195	0.0592
20	0.5735	0.5210	0.0294	0.5551	0.0276	0.5091	0.0775

Table 4: Double Fault Ratios

Dataset	Random Walk		BMTE		BMTE + Pruning	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
1	0.6271	0.0142	0.6857	0.0361	0.7453	0.0608
2	0.6422	0.0162	0.7004	0.0455	0.7161	0.0760
3	0.7805	0.0138	0.8195	0.0295	0.8406	0.0820
4	0.5229	0.0196	0.6458	0.0653	0.6911	0.0662
5	0.1282	0.0024	0.1370	0.0103	0.3606	0.0669
6	0.2321	0.0028	0.2395	0.0094	0.4619	0.0305
7	0.2177	0.0039	0.2263	0.0123	0.4671	0.0332
8	0.2066	0.0032	0.2107	0.0206	0.4421	0.0529
9	0.5866	0.0195	0.7611	0.0907	0.7687	0.1500
10	0.1782	0.0067	0.1783	0.0284	0.4171	0.0487
11	0.3696	0.0087	0.4329	0.1038	0.5633	0.0262
12	0.2427	0.0032	0.2938	0.0369	0.4954	0.0380
13	0.2308	0.0041	0.2408	0.0072	0.4743	0.0313
14	0.2763	0.0025	0.3094	0.0197	0.4989	0.0338
15	0.2389	0.0077	0.2753	0.0200	0.4616	0.0419
16	0.2073	0.0066	0.2524	0.0280	0.3844	0.0753
17	0.2488	0.0060	0.3048	0.0155	0.4783	0.0613
18	0.2892	0.0083	0.3574	0.0335	0.5257	0.0492
19	0.2397	0.0056	0.2593	0.0563	0.4669	0.0604
20	0.2400	0.0049	0.2696	0.0200	0.4670	0.0741

Table 5: Prediction Entropy

Dataset	Random Walk		BMTE		BMTE + Pruning	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
1	0.6473	0.0236	0.5283	0.0607	0.0071	0.0082
2	0.6312	0.0271	0.4982	0.0789	0.0328	0.0136
3	0.4327	0.0269	0.3423	0.0544	0.0164	0.0156
4	0.8078	0.0200	0.6208	0.0922	0.0068	0.0088
5	0.8314	0.0091	0.7612	0.0459	0.0097	0.0133
6	0.9837	0.0015	0.9382	0.0200	0.0163	0.0200
7	0.9661	0.0019	0.8996	0.0299	0.0471	0.0195
8	0.9737	0.0027	0.8807	0.0536	0.0295	0.0282
9	0.7343	0.0252	0.4084	0.1542	0.0351	0.0307
10	0.9299	0.0095	0.8286	0.0705	0.0378	0.0275
11	0.9164	0.0046	0.7884	0.1098	0.0503	0.0426
12	0.9429	0.0053	0.7601	0.0686	0.0642	0.0275
13	0.9776	0.0023	0.9069	0.0274	0.0572	0.0268
14	0.9793	0.0026	0.8798	0.0416	0.0400	0.0299
15	0.9876	0.0015	0.8591	0.0540	0.0562	0.0220
16	0.9574	0.0091	0.7580	0.0821	0.0957	0.0534

Continued on next page

Dataset	Random Walk		BMTE		BMTE + Pruning	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
17	0.9877	0.0036	0.8498	0.0300	0.0623	0.0347
18	0.9780	0.0037	0.8164	0.0620	0.0752	0.0299
19	0.9839	0.0042	0.8247	0.0957	0.1114	0.0321
20	0.9960	0.0107	0.8558	0.0679	0.1064	0.0431

Table 6: Agreement Connectivity

Dataset	Random Walk		BMTE		BMTE + Pruning	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
1	53.97	2.17	58.03	0.02	94.39	8.10
2	53.41	0.60	62.23	0.01	71.07	11.56
3	58.11	3.52	69.27	0.00	85.83	11.23
4	50.30	0.81	55.37	0.02	94.86	9.06
5	51.22	1.08	53.27	0.02	93.17	11.02
6	48.91	0.24	49.03	0.01	85.96	18.63
7	48.79	0.36	49.22	0.04	63.51	16.87
8	48.52	0.64	48.34	0.04	91.08	13.36
9	49.54	0.70	55.12	0.02	74.00	18.58
10	48.78	0.62	50.30	0.05	68.16	20.59
11	50.84	0.87	52.11	0.06	70.92	19.54
12	48.93	1.02	48.01	0.04	62.97	13.83
13	48.50	0.34	48.24	0.03	61.11	18.40
14	48.35	0.31	48.65	0.02	71.30	20.04
15	47.79	0.35	47.06	0.03	59.68	17.28
16	47.18	0.62	45.97	0.04	59.80	14.50
17	48.01	0.36	48.45	0.03	60.72	19.61
18	47.78	0.35	47.93	0.03	56.08	14.60
19	47.57	0.43	46.11	0.08	46.30	3.62
20	46.28	0.83	45.19	0.03	58.69	19.27