Max Markov Chain

by

Mitchell Bucklew

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2022 by the
Graduate Supervisory Committee:

Yu Zhang, Chair
Siddharth Srivastava
Subbarao Kambhampati

ARIZONA STATE UNIVERSITY

December 2022

ABSTRACT

High-order Markov Chains are useful in a variety of situations. However, these processes are limited in the complexity of the domains they can model. In complex domains, Markov models can require 100's of Gigabytes of ram leading to the need of a parsimonious model. In this work, I present the Max Markov Chain (MMC). A robust model for estimating high-order datasets using only first-order parameters. High-order Markov chains (HMC) and Markov approximations (MTDg) struggle to scale to large state spaces due to the exponentially growing number of parameters required to model these domains. MMC can accurately approximate these models using only first-order parameters given the domain fulfills the MMC assumption. MMC naturally has better sample efficiency, and the desired spatial and computational advantages over HMCs and approximate HMCs. I will present evidence demonstrating the effectiveness of MMC in a variety of domains and compare its performance with HMCs and Markov approximations.

Human behavior is inherently complex and challenging to model. Due to the high number of parameters required for traditional Markov models, the excessive computing requirements make real-time human simulation computationally expensive and impractical. I argue in certain situations, the behavior of humans follows that of a sparsely connected Markov model. In this work I focus on the subset of Markov Models which are just that, sparsely connected.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

Markov chains can be used to represent a variety of systems. Ranging from financial markets to human behavior; These robust models use probability values to predict a system's behavior over time and have a wide assortment of applications. Various probability values represent the transition probabilities from one state to another. In this work, we will consider discrete time-homogeneous Markov chains in which every state is connected to every other via two edges (including itself) allowing any state to transition to any other state.

The limitations of Markov chains become clear when applying them to large domains. The space complexity for making predictions grows exponentially. Especially as the order of the model increases, let's say M represents the number of states in a domain and K the order. The space required can be notated as $O(M^K M)$. In this domain, there exist $O(M^K)$ permutations of states, and for each M p-values. For example, if there are 10 states and order 3, a probability table containing 1000 rows and 10 columns must be created to predict every combination of any three states. In complicated domains, there can exist thousands of states, leading to a Markov chain requiring an excessive number of p-values and thus space. A dataset containing 60,000 states of order 3 would require $1.2744e + 19$ parameters. A float is represented using 4 bytes thus this domain would require a CPT table of 864 terabytes. A Markov chain approximation such as the Generalized Mixture Transition Distribution Model (MTDg) can be used to significantly reduce the number of parameters required. However, as shown in the experiment section the time required to calculate these parameters is excessive and unbound.

Our motivation for MMC is to address the limitations of the above models by further imposing parsimony on the model structure to expedite learning and make it scalable to large domains. For the simplest MMC models that we study here, we assume sparse correlations such that only one state (S_{t+l}) in the lags (previous K states) are allowed to influence the current state S_{t+K} . However, MMC allows the influence from S_{t+l} to S_{t+K} to be affected by the presence of the other states, in contrast to the basic MTD models. The MMC has been defined and discussed in detail in the technical report on which this thesis is based on (Zhang and Bucklew, 2022).

In this work, we present a new model able to approximate HMCs using first-order parameters necessitating $O(M^2)$ parameters, the Max Markov Chain. Specifically, we approximate the subset of HMCs with sparse correlations among states. We define the "MMC assumption" as domains in which a state can "generate' a state according to distribution in any future step within the order of the MMC. Where similar models struggle, our proposed model can handle similar datasets in linear time. In the results section, we demonstrate MMC's effectiveness even for extremely large domains.

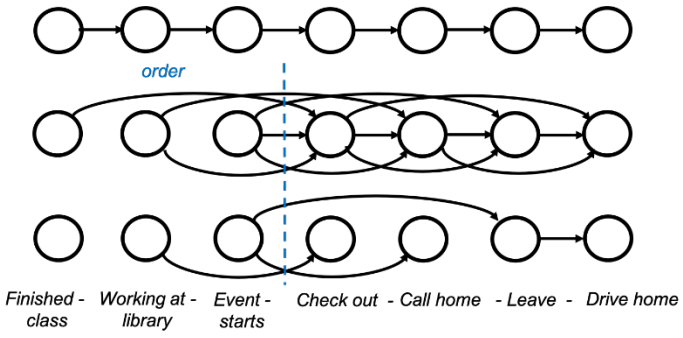

Figure 1: Graphical Representations of, from Top to Bottom, First-order Markov Chain, High-order Markov Chain, and Max Markov Chain. The Order Is 3 for Both MMC and HMC.

In a Markov chain, every state in the previous order states influences the probability of the next. We make the argument that human behavior follows different principles. Our simulation of human multitasking in the results section shows positive results affirming our beliefs. A human naturally switches goals, so not all the previous-order states will be relevant for state prediction. Consider the following example: John arrived at the library and is working on his class assignments in the library. An event suddenly kicks off in the library which is disrupting to John. However, before John leaves, he will need to check out a few textbooks to continue working on the assignments at home. Since he will arrive at home earlier today, he may make a quick call before leaving and driving home. The correlations among these events are shown in Fig. 1, which can be modeled by an MMC. Modeling this scenario with HMC is possible but unnecessary while a first-order Markov chain would be insufficient given the long-term dependencies. Certain types of human behaviors described as both reactive and deliberative may fit the MMC model assumptions well (Schmidt 2000). Such sparse correlations between the states in a process is what we strive to model in this work. An intuitive way to think about an MMC process is to consider a transition system for which any transition may be delayed by an unspecified but bounded amount of time.

1.1 Related Works – Approximate Markov Chain

Markov chain was introduced to model stochastic processes, which has been applied to a diverse set of domains including physics (Randall 2006), computer science (Stewart 1994), geography (Chin 1977), behavior and social sciences (Benjamin 1979). Markov chains rely on the Markov assumption to simplify modeling, learning, and inference. Markov models have also been generalized to enable more expressive-ness and model complexities, such as the Hidden Markov Model (HMM) (Baum and Petrie 1966; Fine, Singer, and Tishby 1998) and factored models (Kearns and Koller 1999). However, Markov chains require an exponential number of parameters in the order of the chain, which makes them in-tractable to maintain for complex domains. This also makes learning very sample inefficient. Approximate HMCs that are more parsimonious and quicker to learn are desired. Popular approximate models (Jacobs and Lewis 1978; Raftery 1985) use auxiliary variables to combine the influences from each of the previous lags for generating the next state. Such models have also been extended to consider model mixtures (Berchtold and Raftery 2002) where the influences to combine are specified with respect to one or multiple lags for better approximations. There are two main limitations of the existing approximate HMC models. While they are more parsimonious than HMCs, learning to optimize the parameters is computationally challenging, often through complex numerical procedures (Raftery 1985; Berchtold and Raftery 2002). Second, learning these models is still inefficient. This is mainly because they do not impose parsimony in model structures, which may introduce overfitting for domains with sparse correlations among the states. This is because, as shown in Fig. 1, not all connections among the states are necessary. Although there are general solutions

for addressing overfitting (Ying 2019) such as parameter regularization, the fundamental problem remains. Max Markov Chain (MMC) addresses these issues by imposing model parsimony while retaining the ability to model long-term dependencies. As we will show later in the discussion, the assumption made in MMC can be gradually relaxed to converge to the full HMC model, resulting in a spectrum of models that are increasing in model complexity. The model structure of MMC may appear like skip-chain sequence models (Galley 2006) and variable-order Markov chains (Roucos, Makhoul, and Schwartz 1982). However, in these prior models, the skipping structures are assumed to be provided a priori or must be learned in a very expensive process. Finally, one may view the sparse correlations among the states as discovering causal relationships (Pearl 2003).

Other works propose Markov model variations capable of dynamic temporal dynamics similarly to MMC (Petropoulos et al., 2017). A key difference being MMC considers a static order whereas the proposed model VDJ-HMM utilizes a variable order process. Additionally, VDJ-HMM is for non-homogeneous MC compared to the time-homogeneous nature of MMCs. Additionally, VDJ-HMM does not follow the assumption a single state generates the next but could be modified to behave similarly to MMC. The domains for each model vary. The temporal dynamics for VDJ-HMM are decided using a first order Markov model. If we represented an MMC using VDJ-HMM a high order Markov model would be required to model the temporal dynamics unnecessarily increasing complexity.

1.2 Related Works – Bayesian Networks

Using a simple Bayesian network (BNs) we can model an MMC as shown below in Figure 2. However, the CPT table for the SGO node would require the same number of parameters as an order 3 Markov chain defeating the purpose of the model. Similarly, our model can be represented as a Hierarchical Bayesian Network (Gyftodimos and Flach, 2002) demonstrating the relationship between the past k states and the SGO. The inference process of a Hierarchical Bayes Nets is the same as a regular BN, in the case there are no loops. Again, leading to an unnecessary number of parameters. Although useful for demonstrating the relationship among states a BN structure is not parsimonious circumventing the advantages of the Max Markov Chain.

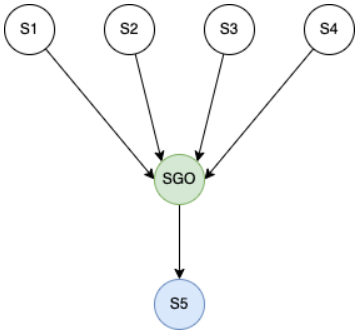

Figure 2: Bayesian Network

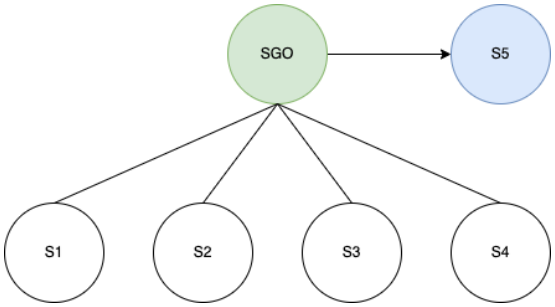

Figure 3: Hierarchical BN

Additionally, an MMC can be represented as a Contingent Bayesian Network (CBN) (Milch, et el. 2005). Such a structure allows us to add conditions under which edges become active as seen in figure 4. Here ">" is used to represent state generation order, and only one edge can be true at a time. We represent the inference process for S5 using a decision tree and the function H(x) which will return true if x is the highest order

state in the previous K (order) states. Using this decision tree format inherit to the CBN, we can represent the entirety of the MMC logic as is shown in Figure 5. This tree would replace the use of a CPT table while also providing the logic for which state to select to influence the outcome. A contingent Bayesian network can be trained similarly to an MMC, and the problem of training condenses down to selecting an SGO. These model structures are useful but do not represent the main innovation presented in MMC, which is the SGO. Our work can be represented as a specific configuration of CBN's as any MMC has an associated CBN. Other work proposes similar concepts whereas edges in Bayesian networks exist only under certain conditions utilizing decision trees (Boutilier et al., 2013). Poole and Zhang utilize *parent contexts* which describes the fact the parents of one variable depend on the values of others. Heckerman et al. proposes a graphical language which similarly allows for the conditional existence of edges. The structure of an MMC can be expressed using these various concepts and should be considered a special case of each. However, in these works it is assumed context is learned through an expensive training process which is in stark contrast to MMC's efficient learning and training process.

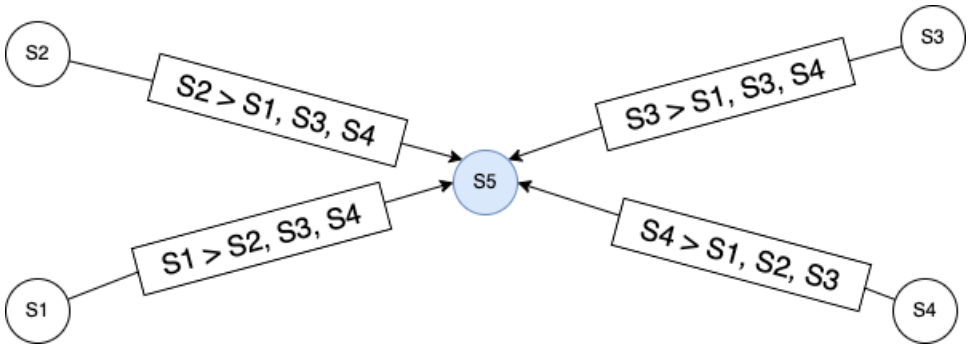

Figure 4: MMC Contingent Bayesian Network of Order K=4

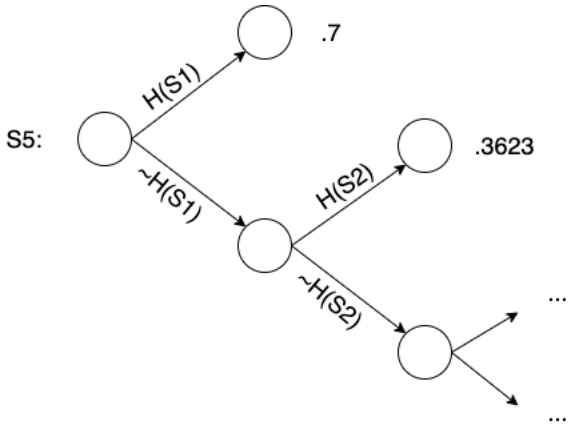

Figure 5: A Decision Tree Showcasing How to Inference S5

CHAPTER 2

MODEL DESCRIPTIONS

Models:	Description:
MTDg	Generalized Mixture Transition Distribution
HMC	High Order Markov Chain
FMC	First Order Markov Chain
MMC	Max Markov Chain

Table 1: Descriptions of Models

2.1 Generalized Mixture Transition Distribution Model (MTDg)

This mixed transition model introduced by Raftery in 1985 is as follows:

$$P(S_{t+K}|S_{t:t+K-1}) = \Sigma_l \lambda_l q_{(t+l)(t+K)}$$

Essentially "a weighted average of transition probabilities from subsequent lags" (Gabryś, 2020). Where $q_{(t+l)(t+K)}$ is a value in an $M \times M$ transition matrix Q ,capturing the influence from state S_{t+l} to S_{t+K}; λ_l is a weight parameter associated with lag l and satisfies $\sum_l \lambda_l = 1$. Hence, an MTD model's parameter size is $M(M-1) + (K-1)$. Even though the MTD model is parsimonious in parameter size, its model structure remains the same as that of HMC. It assumes that the same state at the same lag contributes the same influence on S_{t+K}, regardless of the other states.

2.2 High Order Markov Chains

A discrete-time Markov chain as defined below:

$$P(S_{t+K}|S_{t:t+K-1})$$

Where the current state depends on the previous order (K) states, thus requiring $M^K(M-1)$ parameters where M is the size of the state space. To train the model, we simply generate a CPT table with S^O rows and S columns. We proceed to count the occurrences of each output in the training data given all lags of K states and normalize the rows accordingly resulting in p-values summing to one. During testing, the input is used to select a row in the CPT table and predict the state with the highest p-value. Specifically: $y = \arg\max(P_x)$

2.3 Max Markov Chain

We formally introduce the Max Markov Chain below:

$$P(S_{t+K}|S_{t:t+K-1}) = P(S_{t+K}|S_{t+l^*})$$

Where $l^* = argmax_{S_{t+K},l} P(S_{t+K}|S_{t+l})$. Traditionally a Markov model utilizes all lags (previous order states) to predict the next state. MMC differs because the generation of the next state is dominated by a single previous state. Specifically, the highest order state is used to predict the next based on the SGO. A **State Generation Order** (SGO) is determined based on the training data decided by each state's maximum generation probability. Thus, the highest-order state will have the highest probability value in its

CPT table. In the below example the SGO would be S1 > S2 > S3 based on the maximum generation probability of each state (.8 > .7 > .6).

The highest-order state is selected from the previous order states to predict the next state. Consider the simple CPT table below, S1 is the highest-order state. If the order is 3 and the previous three states are S3, S2, S2 then the probability distribution from S2 will be used to predict the next state. Specifically, we will predict the next state as S2 as that is the highest probability value for the state S2.

State \ Prob	S1	S2	S3
S1	.8	.1	.1
S2	.15	.7	.15
S3	.2	.6	.2

Table 2: Example CPT Table

Importantly an SGO must be selected to maximize the data generation probability as defined below:

$$P(\mathcal{D}_s|\mathcal{M}) = P(\mathcal{D}_s|p_1 : p_m \text{ for state } s) = p_1^{n_1} p_2^{n_2}, ..., p_m^{n_m}$$

We take the log probability in order to avoid underflow:

$$P(D|\mathcal{M}) = n_1 \log p^1 + n_2 \log p^2, ..., n_m \log(p^m)$$

where p_i is the generation probability of state s_i for s and n_i is the number of times in D_s that s_i is generated by s. p_i is the generation probability of state s_i for s and n_i is the number of times in D_s that s_i is generated by s. The optimal SGO has the highest data generation probability out of the set of all possible SGOs. We have developed several techniques for effectively determining the SGO from training data when unknown.

CHAPTER 3

TRAINING MMC MODEL

3.1 Training MMC Process

The training process of an MMC model consists of two steps. First selecting an SGO and second optimizing the parameters. The problem of training an SGO can be expressed by:

$$\max_{\mathcal{M}} P(\mathcal{D}|\mathcal{M}) = \max_{\mathcal{M}} \Pi_d P(d|\mathcal{M})$$

Where $\mathcal{M}$ is the space of all MMC's, and $\mathcal{D}$ represents the training dataset. The goal of training is to select the MMC which maximizes the probability it was used to generate the dataset. Each MMC is differentiated by its SGO. Therefore, the problem becomes selecting the SGO which maximizes the data generation probability.

3.1 Full SGO

In the full SGO identification technique we brute force the solution by iterating through every possible SGO. Of which there exists M! combinations. We generate the data generation probability for every permutation of the state space and order them, accordingly, selecting the highest.

3.2 Greedy SGO

Greedy works by greedily selecting the highest order states individually and sequentially. Each state is considered individually, meaning we count the number of times the

considered state “generates” every other state. We consider each state as its own SGO and calculate the data generation probability as described above.

We count the number of times a state exists in the input per output value then find the product of each count. We perform this process iteratively excluding the states we have already selected into the entire SGO.

3.3 Hillclimb SGO

This approach works by first ordering the states randomly creating a random SGO. Beginning at the highest order state, we consider switching the current state and every subsequent lower order state recalculating the data generation probability. If the switch increases the data generation probability the change is made. This process is repeated until the lowest order state is reached. Although effective, this process is computationally expensive. Each data generation probability calculation requires N iterations. In total this process necessitates $O(S^2 \cdot N)$ iterations.

3.4 SGO Conclusion

We have performed several small-scale experiments to compare the effectiveness of different SGO identification techniques. Although effective, the computational cost of hill climb and full is excessive. Our small-scale experiments showed greedy performing comparable to hill climb. Greedy was effective in identifying the SGO for smaller MMC datasets. Moving forward all experiments and MMC performance demonstrations utilize the greedy SGO identification method. In further work it may be useful to further

compare the various SGO identification techniques. Although the excessive computational cost of hill climbing undermines the advantages of the model.

CHAPTER 4

MMC MISALIGNMENTS

4.1 MMC Misalignment

A misalignment occurs when two states s_x and s_y satisfy that $s_x \succ s_y$ in the given SGO but $p_{x\cdot}^{D} < p_{y\cdot}^{D}$ based on the data. That is, if you order the states in the CPT table by their maximum generation probability and the order is not consistent with the SGO, misalignments exist. We present a technique for addressing these inconsistencies in a dataset.

Misalignments can occur when the SGO is not perfectly identified. It could be due to several reasons, such as a small dataset, inconsistent data, or a dataset simply not following the MMC assumption. Using the greedy method will rarely result in a perfect SGO particularly in exceptionally large state spaces such as 60,000. In some cases, misalignments can occur even when the SGO is accurate, particularly when the training dataset size is low.

Every state can be represented as a node in a graph. Every misaligned state is connected via edges and clusters of misaligned states form subgraphs. Every state in a subgraph must share the same maximum generation probability but not necessarily the most likely generative state.

Given an SGO with a misalignment, the maximum data likelihood is achieved when $p_{x^*} = p_{y^*}$ where p_{x^*} (p_{y^*}) represents the maximum generation probability of state s_x (s_y).

4.2 Combining Misalignments

First compare the SGO to the states ordered by their maximum generation probability and find all inconsistencies. Group the misalignments by combining pairs of misaligned states with commonalities. Once all misaligned subgraphs are created follow the procedure for addressing misalignment graphs:

1. Calculate the new maximum probability value by summing up each maximum probability value and normalizing by the sum of all rows in the sub graphs. **Mp**
2. Calculate the remainder which begins as **1 - Mp**
3. Loop through each state in the sub graph ordered by their maximum generation probability:
    a. Set the most likely state to the new **Mp** value
    b. Distribute the remainder amongst the rest of the states by normalizing the remaining p values and multiplying each by the remainder
    c. If any p values of 0 remain, equally distribute the remainder among them

CHAPTER 5

DATASET DESCRIPTIONS

Dataset:	Description:
HMC	HMC data generated using a randomly created CPT table.
MMC	HMC data generated using the MMC assumption. Generate a first order CPT table randomly, identify the SGO from the CPT and generate the data accordingly.
Casual HMC	HMC data but the appearance of a state in the lag increases its generation probability.
Blocksworld Domain	Data generated using an A* planner in Blocksworld predicate based domain.
Financial Asset	A domain generated from the last 6 months of a financial assets price action.
Watch And Help Simulator	A household simulator designed for human assistant action simulation

Table 3: Dataset Descriptions

5.1 Dataset Introduction

We will now proceed by describing in detail the various datasets used to generate and test our various models. First, we will describe some configuration parameters used to generate these datasets.

Conditional Probability Table (CPT): A table containing rows representing the current state and columns with probability values. A row index represents an agent's current state, and each subsequent column the probability of transitioning to that respective state. Each row sums to 1.

State Space Size: The amount of possible states. **M**

Dataset Size: The total amount of state transitions generated as part of the dataset. **S**

Order: The amount of previous states used to generate the next state. **K**

Now with these definitions in mind we will move forward describing the datasets.

5.2 High Order Markov Chain

The HMC dataset represents a traditional Markov model. A CPT table is randomly generated with M^K rows, each row representing a unique permutation of length K states. A row exists for every permutation of length K out of M states. To generate this table, the current system time is used as a seed necessary for ensuring truly random numbers. The table contains M^K rows but only M columns. Now each value is filled with a random number, and each row is normalized to 1 by summing the contents of each row and dividing each value by its sum.

Once created we can use the CPT table to generate a dataset. We proceed by generating S random sequences of numbers each of length M **.** This represents the input or X component of the dataset. We then reference each corresponding row and sample from its contents to generate each respective output value or y value for every x value.

We then use the sklearn train_test_split to randomly shuffle the data once more and split it into training and testing datasets using the default configuration values.

5.3 Casual HMC

This dataset exists identically to the previously described HMC dataset except for one caveat. Once the CPT is generated, loop through each row and if a state appears in the state combination that row represents, increase its respective generation probability 10-fold. The idea being, if a state is present in the previous K states it increases the likelihood of the state recurring as the output. Once we 10x the values the probability rows are renormalized to sum to one.

5.4 MMC

Expanding on the previous concepts I have described in the HMC, MMC first generates a first order **HMC CPT** table. The **CPT** table is perfectly square containing just as many rows as columns, specifically the shape is $M \cdot M$. A state generation order (**SGO**) is decided by ordering the states' max generation probability.

Although it is similar to HMC, MMC differs in several key aspects. This dataset is a continuous set of values. To generate the dataset random states are randomly generated. We then select the most recent three states and derive from it the highest order state. I will provide a quick example to make the concept clear. If the SGO is $2 > 1 > 3$, the order 3 and the most recent three states are 313, the highest order state in the lags is 1. We will now reference the row representing the 1st state and sample from its contents to generate the next state. The new state is then appended to the first **O** and the process is

repeated with the most recently generated state at the tail. We continue this process until the dataset is of length **DS.** Now the chain is generated we take the first **O** states, assign it as an x value, then assign the **O+1** state as its corresponding y value. Through this process a Max Markov Chain is generated. Given the previous **O** states we can reference the corresponding row and use the according probability values to make a prediction.

5.5 Financial Asset Data

We will get the intraday trading price of each financial asset in various intervals using a financial asset API. Then we will compare the change in price and label them in accordance to how they lie in the distribution using the standard deviation. Then we will use the oldest 85% of data to predict the most recent 15% price action. The motivation here is to develop a model to aid in the day trading of financial assets. Traders can make either long or shorts, to bet that an asset will rise in value or decrease. By obtaining an accuracy of over 50% and spread out over many trades a day trader is likely to turn a profit by using our models.

5.6 Blocksworld

Blocks world is a planning domain in which there are several blocks on a table. Specifically, in our case, it is 4. The blocks start in some random configuration and the planner must reorder the blocks, stacking them accordingly to reach a goal state. Since the domain is deterministic, we have made some changes to make it more appropriate for MMC as well as to simulate human multitasking. There is a single set of actions to lead from a start state to a goal state.

First, we will generate 5,000 random episodes with random start and goal states. We will use an A* planner to generate the solutions to each episode. Then to simulate human multitasking we will concatenate two episodes together. There is a 70% chance to continue working on one episode and a 30% chance it will switch actions to the other episode. Our motivation is as follows: there are two separate blocks world instances on two tables and a human will randomly switch between the two with the probability described above. We randomly select two episodes 2500 times and repeat each pair of episodes 40 times. Each time randomly starting an episode and randomly switching between either one.

5.7 Watch and Help Simulator

The household simulator "Watch and Help Challenge" proposed in "Watch-And-Help: A Challenge for Social Perception and Human-AI Collaboration" r(Puig, 2020) provides a helpful domain for testing human-robot interaction. The original challenge tasks those undergoing it with reproducing human assistance results in a new environment. Agents must first observe a humanesque agent completing tasks alone in an environment, then assist the human in a new environment in completing similar goals. Although interesting, the specifications of the challenge have been modified to support our model.

"Virtualhome" is the domain included as part of the Watch and Help challenge. This domain includes several interior virtual home designs including a variety of items and possible interactions. There are several tasks a humanesque agent can complete in this

domain, such as preparing dinner, cleaning up, and more. Agents can walk around and manipulate items and their states.

The domain includes a Monte Carlo Tree Search Agent (MCTS). Although effective, we do not consider this model an accurate representation of human behavior. In order to simulate humans, we have utilized a finite state machine alongside the MCTS model to create probabilistic goal-changing behavior. There is a certain probability a human will continue their task, and the probability it will randomly move on to another. We hope this state machine models the somewhat inconsistent behavior of humans.

Several features of the domain have been selected in order to accurately represent it using categorical variables. The household domain is defined as a graph of nodes and edges, with edges representing the spatial relationships between the objects as shown in the figure below. This graph format is useful for other types of models but is not compatible with the MMC model and includes unnecessary information. The information provided as input to the MMC Model must be minimized in order to increase performance. For this reason, I have selected several state variables which are all that is necessary for the MMC to make human decisions.

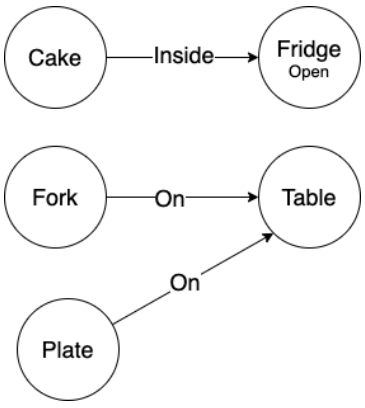

Figure 6: Graphical Representation of Environment Graph

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Experiment Configuration

First, we will present the experiments we conducted using synthetic domains. The experiments provided in the paper utilize three datasets: HMM Data, HMM Casual, and MMC Data.

For each of these datasets we select one parameter to vary on the x axis while keeping the other static: Order (5 static) Varying from 3 - 10, Dataset Size (5k Static) Varying from 2,500 - 20,000 (Step 2,500), State Space Size (7 Static) Varying from 3-15 (Step 1). Any deviations from these ranges are the result of excessive execution times. Each data point on the subsequent 9 graphs represents an average of 30 executions. Each experiment was run on a dedicated machine. Each model was run simultaneously and multithreaded. The specs used for each configuration were: Paperspace C7 Instances including 12 vCPUs and 30GB of RAM. 9 separate machines were used to run each experiment simultaneously. Graphs were generated iteratively. Each line is accompanied by an error band showing the standard deviation of the results.

6.2 Analysis of Results

HMC Data: HMC data is high-order Markov data that satisfies the HMC data generation assumption. Hence, it is expected that MMC would not be able to handle HMC data well. The results are presented in Figs. 3, 5, and 7. First, we can see that HMC model performed poorly even after 20k training samples, illustrating its sample inefficiency. MTD dominated the others in almost all test cases given its smaller parameter size, which makes

it more sample efficient than HMC. However, it also used significantly more time than the other models for training. MTD appeared to be running linearly with respect to the data size but exponentially in the order and state sizes. MMC performed as badly as FMC model (but better than HMC), since each model makes its own assumptions about the data (which do not hold here). Even though the MTD model also makes data assumptions, they seem to be milder, and the model thus fitted better under randomly generated HMC data.

MMC Data: The results with MMC data are presented in Figs. 9, 11, and 13. Results show that MMC indeed outperformed all the other models under MMC data, which is one type of HMC data. What is more interesting was that MTD, again, used significantly more training time than the other models but achieved a performance similar to FMC: it mostly failed to account for the MMC data, which suggested a limitation of MTD models for modeling HMC data. Meanwhile, HMC still suffered gravely from sample inefficiency.

Causal Data: A common type of data that we may frequently encounter is causal data (see our motivating example). Note that causal data only approximately satisfies the MMC data assumptions: it does not restrict that only one lag can generate the current state. Hence, it imposes a challenge for MMC. The results with causal data are presented in Figs. 15, 17, 19. Results show that MMC was able to generalize to this type of data quite well. In general, it outperformed the other models on this type of data. In Fig. 15, you can see that MTD model caught up toward the end as more data was provided, which showed that MMC model was more sample efficient than MTD. This observation was further confirmed

in Fig. 17, and 19, where MTD started comparable to MMC but failed behind MMC as the state size or order size increased (hence more training data would be needed).

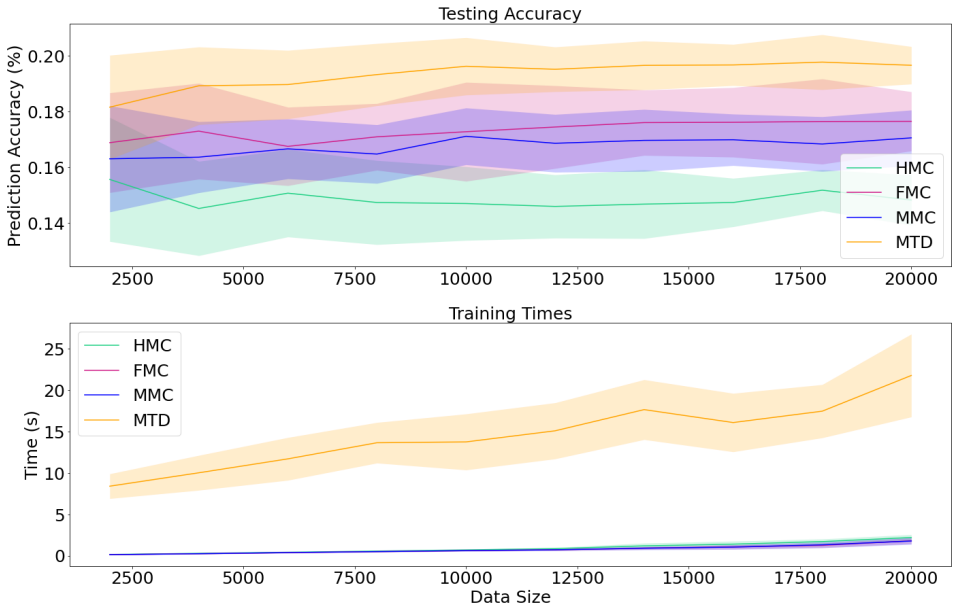

Figure 7: Results for HMC Data While Varying the Data Size. The State Size Is 7 and Order Size Is 5.

MMC & HMC	MMC & FMC	MMC & MTD
0.000	0.284	0.003
0.000	0.883	0.000
0.000	0.010	0.000
0.000	0.031	0.000
0.000	0.806	0.000
0.000	0.037	0.000
0.000	0.089	0.000
0.000	0.011	0.000
0.000	0.026	0.000
0.000	0.174	0.000
0.000	0.186	0.000
0.000	0.006	0.000
0.000	0.065	0.000
0.000	0.001	0.000
0.000	0.018	0.000
0.000	0.122	0.000
0.000	0.310	0.000
0.000	0.083	0.000
0.000	0.016	0.000

Figure 8: HMC - Data Size T-Test

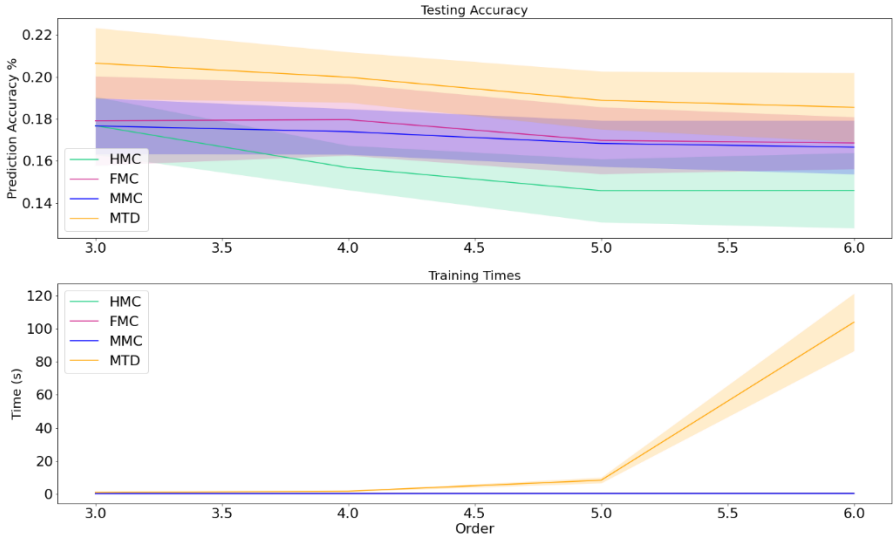

Figure 9: Results for HMC Data While Varying Order Size. The Data Size Is 5k and State Size Is 7.

MMC-HMC	MMC-FMC	MMC-MTD
0.965	0.538	0.000
0.000	0.160	0.000
0.000	0.620	0.000
0.000	0.399	0.000

Figure 10: HMC – Order T-Test

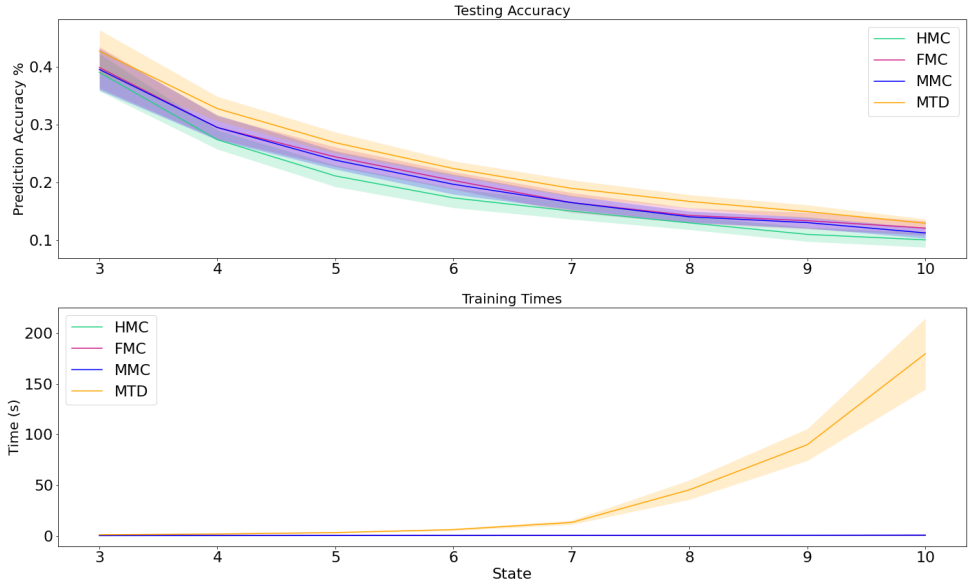

Figure 11: Results for HMC Data While Varying the State Size. The State Size Is 7 and Order Size Is 5.

MMC-HMC	MMC-FMC	MMC-MTD
0.286	0.388	0.000
0.000	0.980	0.000
0.000	0.090	0.000
0.000	0.059	0.000
0.000	0.972	0.000
0.001	0.352	0.000
0.000	0.234	0.000
0.001	0.004	0.000

Figure 12: HMC - State Size T-Test

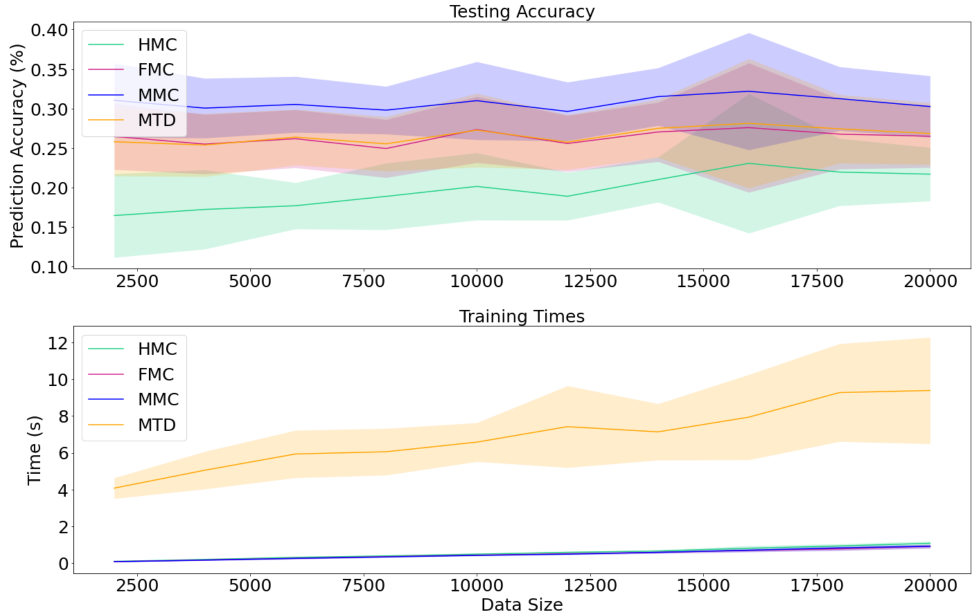

Figure 13: Results for MMC Data While Varying the Data Size.
The State Size Is 7 and Order Size Is 5.

MMC-HMC	MMC-FMC	MMC-MTD
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

Figure 14: MMC - Data Size T-Test

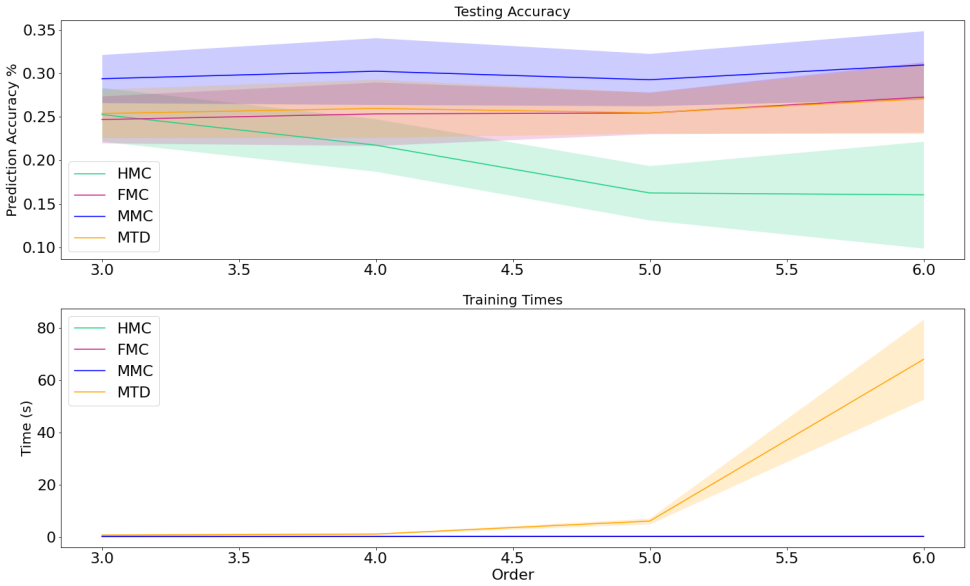

Figure 15: Results for MMC Data While Varying Order Size. The Data Size Is 5k and State Size Is 7.

MMC-HMC	MMC-FMC	MMC-MTD
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

Figure 16: MMC - Order Size T-Test

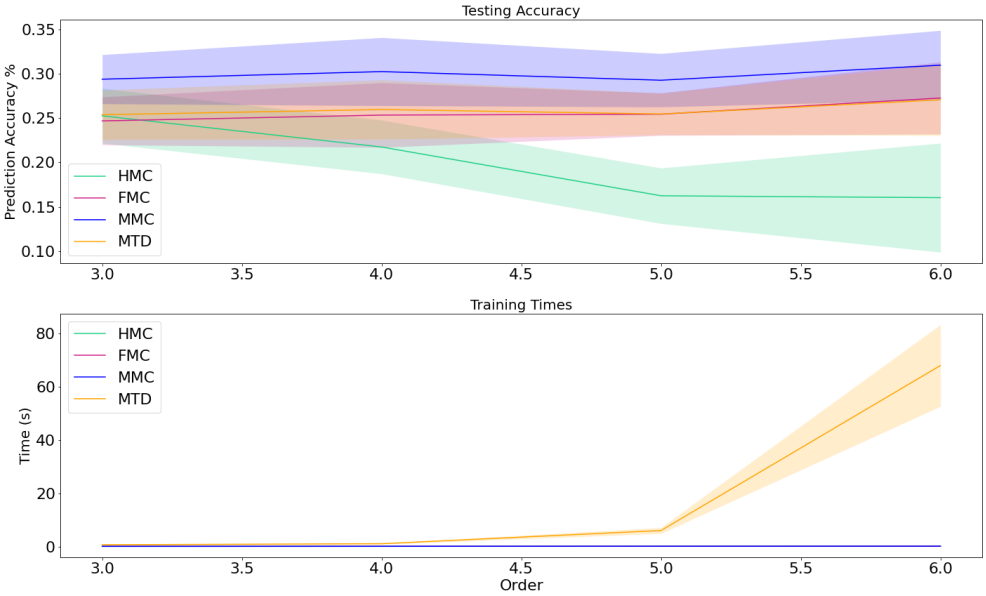

Figure 17: Results for MMC Data While Varying the State Space Size. The Data Size Is 5k and Order Size Is 5.

MMC-HMC	MMC-FMC	MMC-MTD
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

Figure 18: MMC - State Size T-Test

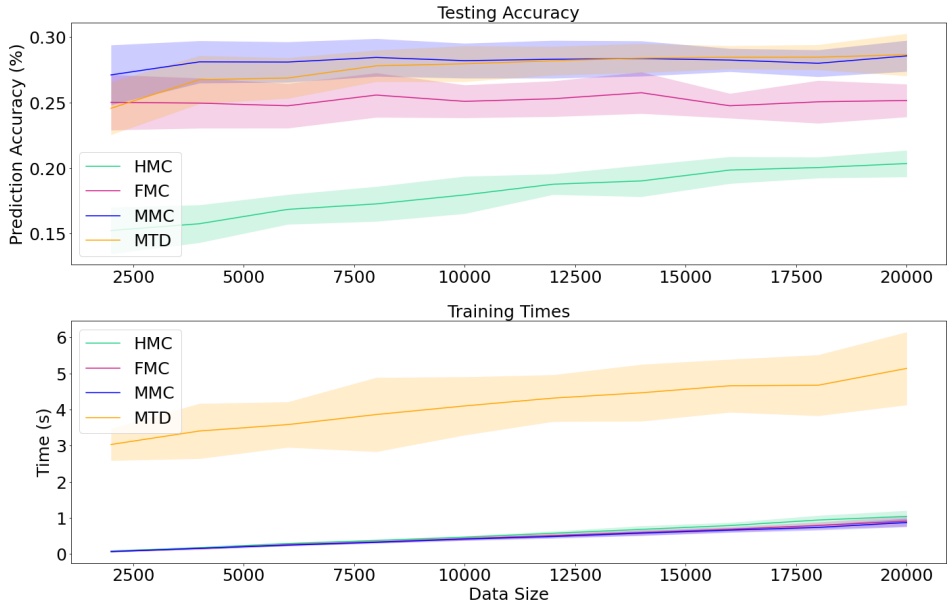

Figure 19: Results for causal data while varying the data size. The state size is 7 and order size is 5.

MMC-HMC	MMC-FMC	MMC-MTD
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.009
0.000	0.000	0.000
0.000	0.000	0.049
0.000	0.000	0.025
0.000	0.000	0.379
0.000	0.000	0.002
0.000	0.000	0.824
0.000	0.000	0.540
0.000	0.000	0.149
0.000	0.000	0.081
0.000	0.000	0.283
0.000	0.000	0.601
0.000	0.000	0.056
0.000	0.000	0.404
0.000	0.000	0.197
0.000	0.000	0.015
0.000	0.000	0.283

Figure 20: Causal - Data State Size T-Test

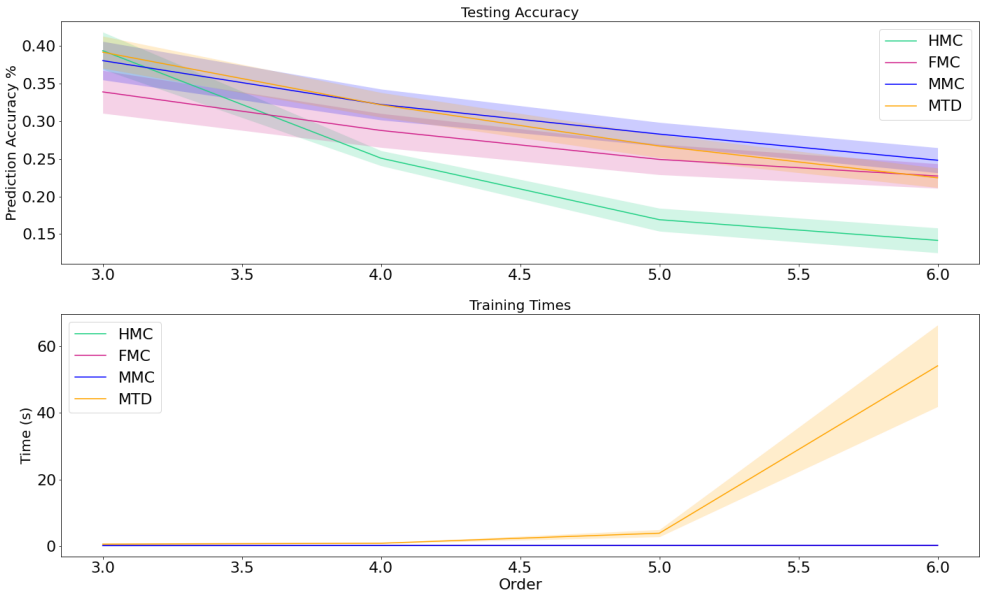

Figure 21: Results for Causal Data While Varying Order Size. The Data Size Is 5k and State Size Is 7.

MMC-HMC	MMC-FMC	MMC-MTD
0.007	0.000	0.006
0.000	0.000	0.901
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

Figure 22: Causal - Order Size T-Test

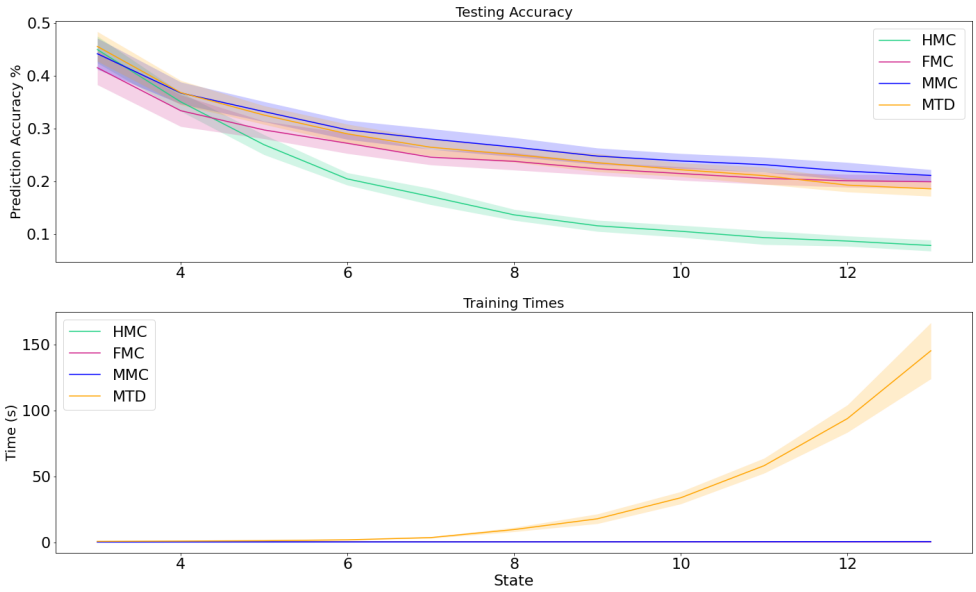

Figure 23: Results for Causal Data While Varying the State Space Size. The Data Size Is 5k and Order Size Is 5.

MMC-HMC	MMC-FMC	MMC-MTD
0.148	0.000	0.015
0.001	0.000	0.909
0.000	0.000	0.116
0.000	0.000	0.058
0.000	0.000	0.001
0.000	0.000	0.000
0.000	0.000	0.006
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.001	0.000
0.000	0.004	0.000
0.000	0.001	0.000

Figure 24: Causal - State Size T-Test

6.3 Blocksworld Results

The results below show MMC greatly outperforming FMC. Although it is important to note the changes which have been made to the domain. Blocksworld is traditionally deterministic and does not follow the MMC assumption unless heavily engineered to be suitable for the MMC model. The configuration for this domain simulates a goal changing situation which is common for human behavior. In the case the goal remains static MMC performs poorly, as given one state it is easy to determine the immediate next state. Our changes enforce the sparse correlation among states required for MMC to perform well. The two other models used for testing, HMC & MTDg were naturally unable to handle the extremely large state size. In our previous experiments, the state space size never exceeded 20. Our blocksworld domain includes over 200,000 different states.

Episode Size: 100000
State Size: 205105
Order: 3

Noise:	MMC Testing (%)	MMC Action Prediction (%)	MMC Training (s)	FMC Testing (%)	FMC Action Prediction (%)	FMC Training (s)
0%	21.44	18%	15.66	1.44	3	.29
20%	10.88%	13%	12.91s	1.12	4	.21

Table 4: Blocksworld Results while Varying Noise

6.4 Financial Data Results

These results bode well for the usefulness of the model, as this is real world data extracted from the price action of various financial assets such as Bitcoin, Ethereum, and Microsoft stock prices. In all cases MMC performs better or comparable to all other models. The idea for this dataset was taken from the open source MTDg implementation we used (Gabrys, 2020). Although instead of using the outdated USD to EUR data Gabrys provided, we used recent real-world data from a variety of assets to demonstrate the versatility of the model. Below are the results:

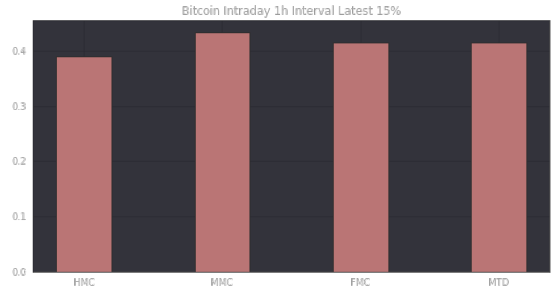

Figure 25: Bitcoin Intraday Price Action Using Latest 15% of Data for Testing Order 5

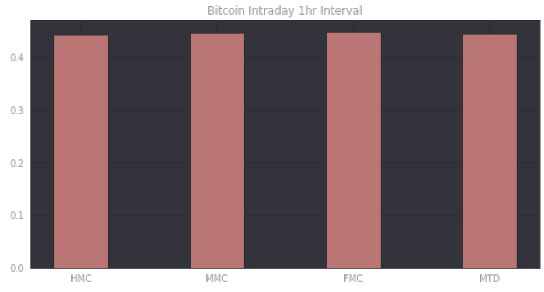

Figure 26: Bitcoin Intraday Price Action Split Randomly Order 5

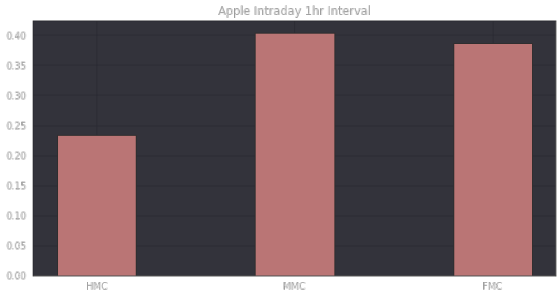

Figure 27: Apple Intraday Price Action Split Randomly 1hr Interval Order 5

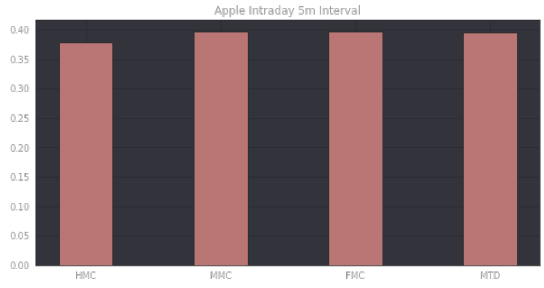

Figure 28: Apple Intraday Price Action Split Randomly 5m Interval Order 5

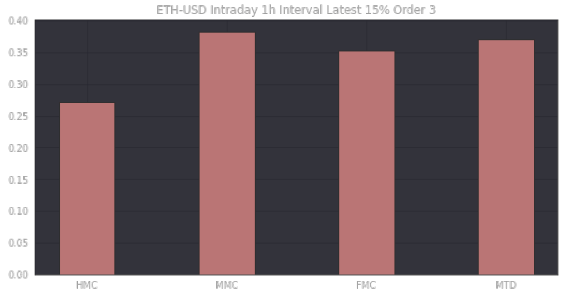

Figure 29: ETH Intraday Price Action 1hr Interval Latest 15% Order 3

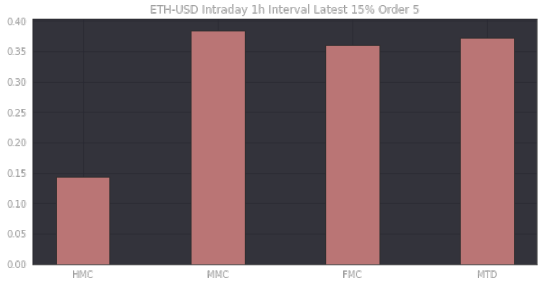

Figure 30: ETH Intraday Price Action1hr Interval Latest 15% Order 5

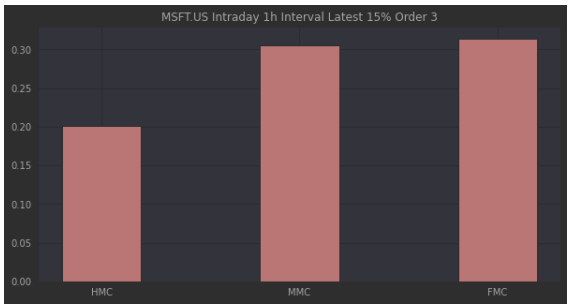

Figure 31: MSFT Intraday Price Action 1hr Interval Latest 15% Order 3

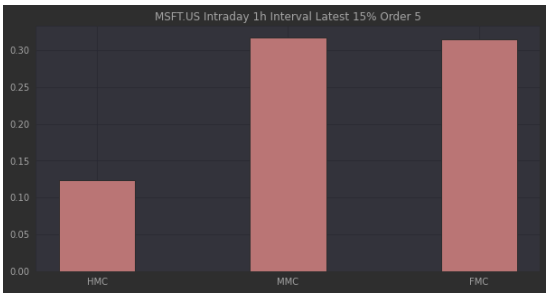

Figure 32: MSFT Intraday Price Action 1hr Interval Latest 15% Order 5

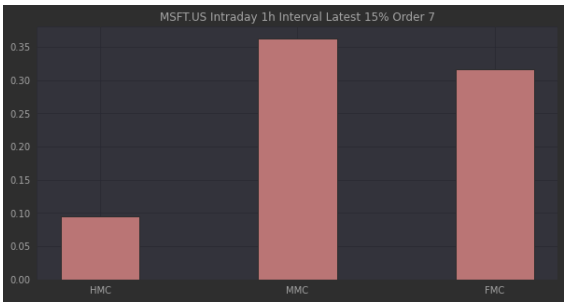

Figure 33: MSFT Intraday Price Action 1hr Interval Latest 15% Order 7

6.5 Watch and Help Results

Below are the results for the watch and help domain. These results are less than ideal but due to insufficient training data. The simulator is slow, so it is computationally expensive to generate training data. Each state occurs about 4 times which is not enough to compute accurate probability values. In further work it would be beneficial to develop

a more complex human simulator able to change goals in a less primitive manner. Also, the tasks designed for the simulator involve merely relocating objects around the environment. This results in short plans not suitable for this application.

Dataset Size: 28174

State Count 8038

Order 5

Noise:	MMC Testing (%)	MMC Training (s)	FMC Testing (%)	FMC Training (s)
0%	0.67	1027s	0.08%	1.18

Table 5: Watch and Help State Prediction Results

CHAPTER 7

CONCLUSION

In this paper, we introduced the Max Markov Chain (MMC) as a novel model for stochastic processes. The motivation was to construct an efficient model that enforced parsimony in model structure to model a subset of high-order processes that were useful. The simple model structure also enabled the model to scale to large domains. We provided an analytical solution for parameter estimation and formally proved it being a local maximum. Approximate solutions were presented based on hill climbing and a greedy heuristic. Results verified that MMC was efficient at handling MMC data, able to generalizing to causal data (which is a common type of data), and scalable to large domains. Results also hinted on other domains (e.g., multi-agent task planning and human-robot interaction) where MMC would be expected to excel.

7.1 Applications of MMC

In domains necessitating collaboration an important challenge is the ability of an agent to maintain models of others. In a partially observable environment such as many real-world situations agents have limited knowledge of each other and must observe each other's actions in order to predict their future behavior and provide the most optimal assistant action (Zhang, et al 2015). In Collaboratory environments, frequently agents are not able to observe the complete plan trace of other agents. Consider the example of multiple people working at a restaurant to serve meals. If a cook is in the back, he may only be able to observe the actions of the waiting staff while they are in the kitchen. He must observe how they act in the kitchen to predict their actions on the restaurant floor.

Models of other agents may be expensive to train and require large amounts of complete data. Here MMC can be used to quickly generate models of other agents based on their actions using little, noisy, or incomplete training data.

In real world domains with finite resources, it is important for robot agents to be able to accurately model other agents to avoid conflicts (Chakraborti, et al 2016). This behavior is required for robotic assistants to be considered "socially acceptable". Robot agents must have some sort of predictive ability in order to not hinder humans and be more obnoxious than helpful. Consider a robot is cleaning up the kitchen while a human is making coffee. While the mug sits out on the counter, the robot must be able to predict the human's imminent interaction with it to avoid putting it back in the cabinet.

In collaboratory environments, agents can use an MMC to predict other agent's actions in real time in order to provide the most optimal assistive action. Models may have expensive inference processes incapable of the instant prediction required for human interactions. Due to its parsimonious structure MMC can make predictions extremely quickly.

There may even be cases where interrupting a human's goal can be beneficial thus increasing "Stigmergy"; the ability of agents to improve the efficiency of their plans based on the changes to the environment made by the actions of other agents (Chakraborti, et al. 2016). Particularly in the case that agents are coinhabitants of the same environment as described by Chakraborti and others. MMC can be used not only to avoid an interruption of a plan but to strategically interrupt other agents plans. Perhaps in the case a robot agent has knowledge not known by a human and can interrupt their plans to reveal such information and increase serendipity.

Additionally, it is crucial in human-robot teaming circumstances for a robot's actions to be understood by humans (Gong and Zhang, 2018). Virtue signaling can be used by robot assistants in order to signal their intentions and preemptively explain their plans. A robot may generate natural language in order to communicate its plan (Gong and Zhang, 2018), visual cues (Andersen, et al 2016), or legible motions (Dragan, 2013). In order to generate comprehensible plans a robot must model a human's expectations of its own behavior. The proposed CRF model for this purpose can be replaced using an MMC. For example, we can ask a human how they expect a robot to behave to generate plan traces. Using these plan traces we can create an MMC and supply it to a robot assistant in order to model the human's expectations. If the robot's plan does not match the human's expectations, it may be inexplicable and virtual signaling will fail.

REFERENCES

Andersen R. S.; Madsen O.; Moeslund T. B.; and Amor H. B. 2016. Projecting robot intentions into human environments. In RO-MAN. 294–301.

Baum, L. E.; and Petrie, T. 1966. Statistical inference for probabilistic functions of finite state Markov chains. The annals of mathematical statistics, 37(6): 1554–1563.

Benjamin, L. S. 1979. Use of structural analysis of social behavior (SASB) and Markov chains to study dyadic interactions. Journal of Abnormal Psychology, 88(3): 303.

Berchtold, A.; and Raftery, A. 2002. The mixture transition distribution model for high-order Markov chains and non-Gaussian time series. Statistical Science, 17(3): 328–356.

Boateng, A.; and Zhang, Y. 2021. Virtual shadow rendering for maintaining situation awareness in proximal human-robot teaming. In Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction (pp. 494-498).

Boutilier C; Friedman, N; Goldszmidt, M; and Koller D; 1996. Context-specific independence in Bayesian networks. In Proc. 12th UAI, pages 115–123.

Chakraborti, T.; Briggs, G.; Talamadupula K.; Zhang Y.; Scheutz, M.; Smith D.; and Kambhampati S.; 2015. Planning for Serendipity in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Chakraborti. T.; Zhang, Y; and Kambhampati, S.; 2016. Planning with Resource Conflicts in Human-Robot Cohabitation in International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

Chin, E. H. 1977. Modeling daily precipitation occurrence process with Markov chain. Water resources research, 13(6): 949–956.

Dragan A.; and Srinivasa S. 2013. Generating Legible Motion. In Robotics: Science and Systems

Gabryś, P. (n.d.). *MTD-learn/2020-07-06-mixture-transition-distribution.MD at master · PIOTREKGA/MTD-Learn*. GitHub. Retrieved October 28, 2022, from https://github.com/PiotrekGa/mtd-learn/blob/master/post/2020-07-06-Mixture-Transition-Distribution.md

Galley, M. 2006. A skip-chain conditional random field for ranking meeting utterances by importance.

Gong Z.; and Zhang Y. 2018. Behavior Explanation as Intention Signaling in Human-Robot Teaming in Proceedings of IEEE International Conference on Robot and Human Interactive Communication (RO-MAN).

Gong. Z.; and Zhang, Y.; 2018. HRI Workshop on Explainable Robotic Systems, Robot signaling its intentions in human-robot teaming

Gyftodimos, E; and Flach, P. 2002. Hierarchical Bayesian Networks: A Probabilistic Reasoning Model for Structured Domains.

Heckerman D; Meek, C; and Koller D. 2004. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research.

Jacobs, P. A.; and Lewis, P. A. 1978. Discrete time series generated by mixtures. I: Correlational and runs properties. Journal of the Royal Statistical Society: Series B (Methodological), 40(1): 94–105.

Kearns, M.; and Koller, D. 1999. Efficient reinforcement learning in factored MDPs. In IJCAI, volume 16, 740–747.

Milch, B.; Marthi, B.; Sontag, D.; Russell, S.; Ong, D. L.; and Kolobov, A. 2005Approximate inference for infinite contingent Bayesian networks. In International Workshop on Artificial Intelligence and Statistics (pp. 238-245). PMLR

Pearl, J. 2003. Statistics and causal inference: A review. Test, 12(2): 281–345.

Poole and Zhang. Exploiting contextual independence in probabilistic inference. JAIR, 18:263–313, 2003.

Petropoulos, A; Chatzis, P; and Xanthopoulos, S; 2017. A hidden Markov model with dependence jumps for predictive modeling of multidimensional time-series, Information Sciences.

Raftery, A. E. 1985. A model for high-order Markov chains. Journal of the Royal Statistical Society: Series B (Methodological), 47(3): 528–539.

Randall, D. 2006. Rapidly mixing Markov chains with applications in computer science and physics. Computing in Science & Engineering, 8(2): 30–41.

Roucos, S.; Makhoul, J.; and Schwartz, R. 1982. A variable- order Markov chain for coding of speech spectra. In ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 7, 582–585. IEEE.

Schmidt, B. 2000. The modelling of human behavior, volume 132. Society for Computer Simulation International.

Stewart, W. J. 1994. Introduction to the numerical solution of Markov chains. Princeton University Press

Ying, X. 2019. An overview of overfitting and its solutions. In Journal of physics: Conference series, volume 1168, 022022. IOP Publishing.

Zhang, Y; Sreedharan, S; and S. Kambhampati. 2015. Capability Models and Their Applications in Planning in International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

Zhang, Y; and Bucklew, M. 2022. Max Markov Chain from https://arxiv.org/abs/2211.01496

Zhang, Y.; Zhuo H. H.; and Kambhampati, S. 2015. Plan explainability and predictability for cobots. CoRR abs/1511.08158.