Finite Markov Mixture Model for Automatic Parameter Tuning

Freddy Chong Tat Chua
School of Information Systems
Singapore Management University
freddy.chua.2009@smu.edu.sg

1. INTRODUCTION

1.1 Motivation

Combinatorial Optimization problems occur frequently as a fundamental problem in operations research. The optimal solution to many problems in operations research can be derived from the optimal solution of basic combinatorial problems. Some examples of such problems include Traveling Salesman Problem (TSP), Quadratic Assignment Problem (QAP), and Job Shop Scheduling Problem (JSP). Unfortunately, it has been theoretically proven that these combinatorial problems are NP Complete and no known algorithm exists that finds the solution in polynomial time. Existing techniques exist to provide sub-optimal solutions in reasonable amount of time. One such technique is the Iterated Local Search (ILS) [10].

1.2 Iterated Local Search

The main purpose of Iterated Local Search (ILS) is to provide an intelligent principle of searching for a locally optimal solution among the space of all solutions. There are several basic principles for performing ILS,

1. Finding a good initial solution
2. Iteratively approach a local optimum
3. Perturbate a solution s_1 to another solution s_2 such that an iterative method defined above will not allow us to go from s_1 to s_2.
4. Restart the search to explore different areas of the solution space.

To perform these methods, a user running an ILS algorithm for solving a combinatorial problem will have to decide parameters for these methods. These parameters influence the search pattern of ILS. For example, which strategy to use for iteration, how far each perturbation should be, how many times to restart the search and how to construct the initial solution. Because of the NP-hardness of these combinatorial problems, none of the heuristic or parameter settings is generalizable to all instances of combinatorial problems.

1.3 Automatic Parameter Tuning

As described earlier, there is no single parameter setting that works well for all instances of a combinatorial problem. The user who utilizes heuristics will run ILS several times while varying the parameters to observe the best parameter settings. Such efforts are laborious and Automatic Parameter Tuning (APT) approaches have been proposed to automate the tuning of such parameters [1]. The APT algorithm involves running the set of instances with some initial parameter settings. The output from an APT algorithm is used as a feedback to re-adjust the parameters. Several runs of such tuning occurs and the APT algorithm then determines the best set of parameter settings.

One shortcoming of APT is that it produces a single set of parameter settings for all given instances. Clearly this is an overly optimistic assumption because problem instances differ from one another, hence, parameter settings which work for some instances may not work for others. Earlier work had proposed that instances can be divided into clusters where intra-cluster instances have high similarity while inter-cluster instances have low similarity [3]. APT may then be applied to each cluster independently of other clusters.

1.4 Generic Instance-Based Clustering

In general, we want to use a clustering approach for APT such that the clustering is generalizable to all combinatorial problems. That is given an algorithm $\mathcal{A}$ where $\mathcal{A}$ can be any ILS algorithm for combinatorial problem, and a problem instance $\mathcal{I}$ where $\mathcal{I}$ is a combinatorial problem that $\mathcal{A}$ solves, the clustering technique should apply to all of such $\mathcal{A}$ and $\mathcal{I}$. Such generalized clustering approach had been proposed [7]. The proposed idea in [7] uses the search trajectory patterns of an algorithm $\mathcal{A}$ to produce a sequence that acts as features of an instance $\mathcal{I}$. Since the sequence is produced by a generalized ILS algorithm $\mathcal{A}$ and not from the instance itself, then any combinatorial problem that uses algorithm $\mathcal{A}$ is suitable for the APT approach we described here. The description of search trajectory is given in [4].

A clustering algorithm for sequences requires the measurement of distance between two sequences. The sequence distance metric used in [7] is highly dependent on the length of the sequence. However, the length of the sequence is not a representative feature of the problem instance. Hence, we propose to use a Markov Mixture Model that is able to clus-

ter sequences without much consideration for their length.

1.5 Overview of Automatic Parameter Tuning using Clustering

We summarized the APT method as follows.

1. For existing N problem instances $\mathcal{I}$, run algorithm $\mathcal{A}$ to obtain their search trajectory sequences $S = (s_1, s_2, \ldots, s_N)$ using Table 1.

2. Cluster the sequences S into K clusters for which K is to be determined automatically by the clustering algorithm.

3. For each cluster k, run CALIBRA to determine a suitable set of parameters.

4. Subsequently for any arbitrary instance i that does not belong to the original N problem instances, run algorithm $\mathcal{A}$ to obtain sequence s_i.

5. Assign a cluster k to s_i and use the parameter settings determined by CALIBRA earlier for cluster k to solve i.

Table 1: Position Types of Solution

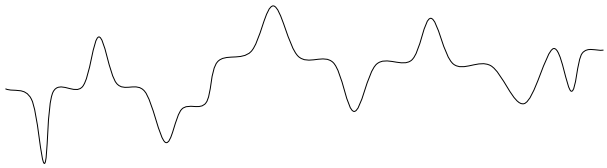
Position Type	Symbol	<	=	>
SLMIN (strict local min)	S	+	-	-
LMIN (local min)	M	+	+	-
IPLat (interior plateau)	I	-	+	-
SLOPE	P	+	-	+
LEDGE	L	+	+	+
LMAX (local max)	X	-	+	+
SLMAX (strict local max)	A	-	-	+

1.5.1 Example

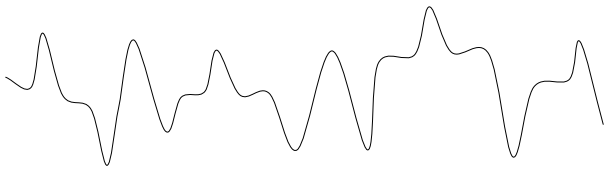
Suppose we have the following functions $f_1(x), f_2(x)$ and $f_3(x)$ as shown in Figures 1(a), 1(b) and 1(c). Since the functions are not convex, finding the minimum value of the functions is NP-hard. We can use an algorithm $\mathcal{A}$ to search for the optimal values and solution of the functions. From the figures, $f_1(x)$ and $f_2(x)$ share more similarities compared to $f_3(x)$. Hence, we hypothesize that the parameters for algorithm $\mathcal{A}$ should use similar values for solving $f_1(x)$ and $f_2(x)$ while $f_3(x)$ should use another set of parameters. i.e. $f_1(x)$ and $f_2(x)$ belongs to the same cluster of problem instance and $f_3(x)$ is in a separate cluster.

In order to express the pattern of the sequences in a computer recognizable form, we use the search positions in Table 1 returned by algorithm $\mathcal{A}$. Figures 2(a), 2(b) and 2(c) shows how we reduce the problem instances into sequences. Notice that in the sequence of $f_3(x)$ the state after a local min/max is SLOPE (P). This is an indication of how different $f_3(x)$ is from $f_1(x)$ and $f_2(x)$.

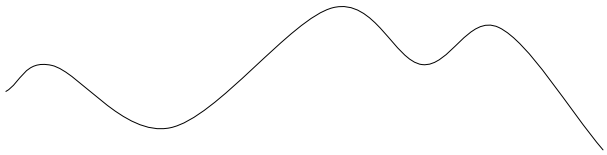
We do not restrict this method only to finding minimum points on functions. The method is generalizable to any combinatorial problem if algorithm $\mathcal{A}$ is able to generate these states. In general, it is not difficult to obtain such sequences from $\mathcal{A}$.



(a) $f_1(x)$

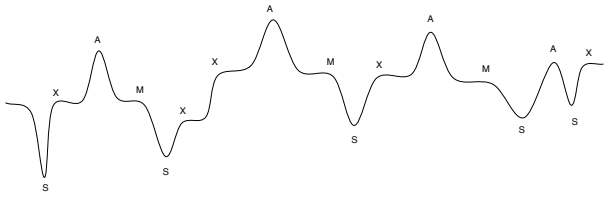


(b) $f_2(x)$



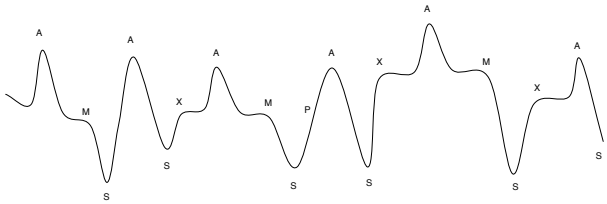
(c) $f_3(x)$

Figure 1: Example of Instances



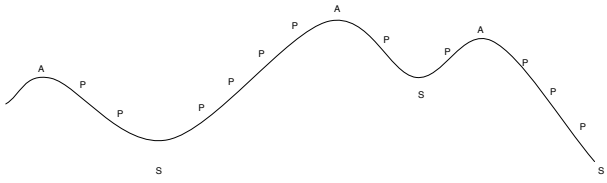
S X A M S X X A M S X A M S A S X

(a) Sequence of $f_1(x)$



A M S A S X A M S P A S X A M S X A S

(b) Sequence of $f_2(x)$



A P P S P P P P A P S P A P P P S

(c) Sequence of $f_3(x)$

Figure 2: Example of Instances reduced to Sequences

1.6 Clustering using Bayesian Networks and Markov Process

From here onwards, we will describe the Finite Markov Mixture Model for sequence clustering. The closely related Hidden Markov Model is already in used in Bioinformatics for DNA sequence clustering [11]. Markov Model is a special case of directed graphical models, hence, it is a form of bayesian network. We will show detailed equations regarding the approximate inference procedure. Because there is no domain expert to inform us about how the sequences should be clustered, hence, we do not have prior information about the conditional probabilities of the bayesian network. We will have to learn the conditional probability tables from the sequence data which is known as parameter learning. Parameter learning is often perform together with probabilistic inference. My description here extends the lecture of Markov Decision Process and Bayesian Network.

Infact, the model described here represents the current state of the art in Statistical Machine Learning. The description on Finite Markov Mixture Model proceed as follows, we first explain the basic Markov Model. Then we show that Markov Model can be generalized by Finite Markov Mixture Model [6].

For a brief overview of how the entire algorithm works, readers may want to skip the derivations and skip to the Appendix.

2. MARKOV MODEL

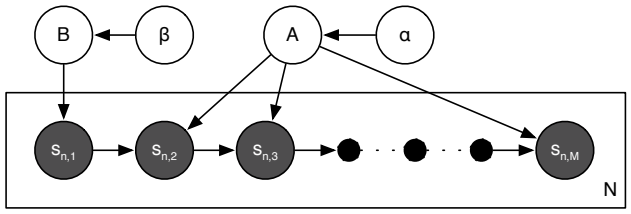


Figure 3: Markov Model

Refer to Figure 3 for a graphical representation of Markov Model. The shaded circles represent observed random variables while unshaded circles represent latent variables that we intend to learn from data. The rectangular box encapsulates N repetitions of each sequence s_n. The directed edges represent causal relationships.

2.1 Basics

Suppose we have a list of N sequences $S = (s_1, \ldots, s_N)$. Each sequence s_n of length M is a string of W discrete states. We let $s_{n,m} \in \{1, 2, \ldots, W\}$ denote the state at the m^{th} position in sequence s_n. Using the markov assumption, the next state $s_{n,m+1}$ depends only on $s_{n,m}$ and state transition probabilities [5]. That is we can state that,

$$P(s_{n,m+1}|s_{n,m}, s_{n,m-1}, \ldots, s_{n,1}, A, B) = P(s_{n,m+1}|s_{n,m}, A)$$

where A is the state transition probabilities matrix.

We can define a markov model with two set of parameters. The initial state probability vector $B \in \mathbb{R}^{\mathbb{W}}$ and the state transition probability matrix $A \in \mathbb{R}^{\mathbb{W}\times\mathbb{W}}$. Formally, the markov model can be described as follows,

$$P(s_{n,1} = i|B) = b_i \tag{1}$$

$$P(s_{n,m+1} = j|s_{n,m} = i, A) = a_{i,j} \tag{2}$$

where $1 \leq i \leq j \leq W$, b_i is the i^{th} element in B and $a_{i,j}$ is the (i^{th}, j^{th}) element in A. Equation 1 represents the probability that a sequence s_n has state i as the initial state. Equation 2 represents the probability that a sequence s_n with state i in the m^{th} position will move to state j in the $m + 1^{th}$ position.

2.2 Parameter Learning

Now that we have a basic idea of markov model, we will now investigate how to obtain the values of B and A from a list of N sequences S. Before showing the bayesian method of learning these parameters, we will like to indicate that learning B and A is a numerical optimization problem. Suppose the values in B and A compresses our knowledge of all these sequences. We can specify the sequences S as follows,

$$P(s_1, \ldots, s_N|B, A) = \prod_{n=1}^{N} P(s_n|B, A) \tag{3}$$

$$= \prod_{n=1}^{N} P(s_{n,1}|B) \prod_{m=1}^{M} P(s_{n,m+1}|s_{n,m}, A) \tag{4}$$

Traditional numerical optimization techniques can obtain B and A using first order differentiation. However, to lay the ground work for nonparametric bayesian statistics, we will show the bayesian method of optimization.

We will approximate $B = (b_1, b_2, \ldots, b_W)$ using a dirichlet distribution with hyper-parameters $\beta = (\beta_1, \beta_2, \ldots, \beta_W)$.

$$(b_1, \ldots, b_W) \sim Dir(\beta_1, \ldots, \beta_W) \tag{5}$$

$$P(b_1, \ldots, b_W|\beta_1, \ldots, \beta_W) \propto \prod_{i=1}^{W} b_i^{\beta_i - 1} \tag{6}$$

$$E(b_i|\beta_1, \ldots, \beta_W) = \frac{\beta_i}{\sum_{j=1}^{W} \beta_j} \tag{7}$$

Equation 5 is the notation for denoting that B follows a dirichlet distribution. Equation 6 is the probability density function for dirichlet distribution. Equation 7 is the expected value for each parameter of the dirichlet distribution.

We will now show how to learn $(b_1, \ldots, b_W)$ from the first element of the N sequences. Suppose we observe the first element of sequences S and decide to update our belief for $(b_1, \ldots, b_W)$.

$$P(b_1, \ldots, b_W|s_{1,1}, \ldots, s_{N,1}, \beta_1, \ldots, \beta_W) \tag{8}$$

$$= \frac{P(s_{1,1}, \ldots, s_{N,1}|B)P(B|\beta)}{P(s_{1,1}, \ldots, s_{N,1})} \tag{9}$$

$$\propto P(b_1, \ldots, b_W|\beta_1, \ldots, \beta_W) \prod_{n=1}^{N} P(s_{n,1}|b_1, \ldots, b_W) \tag{10}$$

$$\propto P(b_1, \ldots, b_W|\beta_1, \ldots, \beta_W) \prod_{i=1}^{W} b_i^{c_i} \tag{11}$$

$$\propto \prod_{i=1}^{W} b_i^{c_i + \beta_i - 1} \tag{12}$$

where c_i is the number of times state i appear as first element of sequences S. Equation 8 denotes the result of updating B given the sequences S. Equation 9 is a result of Bayes Theorem. Equation 10 replaces the denominator of Equation 9 with a normalizing constant. Equation 11 is the result of replacing $P(s_{n,1}|B)$ using Equation 1. Equation 12 replaces $P(B|\beta)$ using Equation 6. Note that Equation 12 has the same form as Equation 6, this indicates that updating such distributions are simple counting operations.

We will now show how to learn each row of the transition matrix A. Suppose each row A_i of the matrix A is approximated using a dirichlet distribution. The formulation for parameter learning is very much similar to learning B. Hence, we will just provide the final parameter update equation.

$$A_i \sim Dir(\alpha_{i,1}, \ldots, \alpha_{i,W})$$

$$P(A_i|\alpha_i) \propto \prod_{j=1}^{W} a_{i,j}^{\alpha_{i,j}-1}$$

$$\begin{aligned} P(A_i|S,\alpha_i) &= \frac{P(S|A_i)P(A_i|\alpha_i)}{P(S)} \\ &\propto P(A_i|\alpha_i) \prod_{n=1}^{N}\prod_{m=1}^{M} P(s_{n,m+1}|s_{n,m}=i, A_i) \\ &\propto \prod_{j=1}^{W} a_{i,j}^{d_{i,j}+\alpha_{i,j}-1} \end{aligned}$$

where $d_{i,j}$ is the number of times state j appear after state i. Note that the updating of A_i is again a simple counting operation.

3. MARKOV MIXTURE MODEL

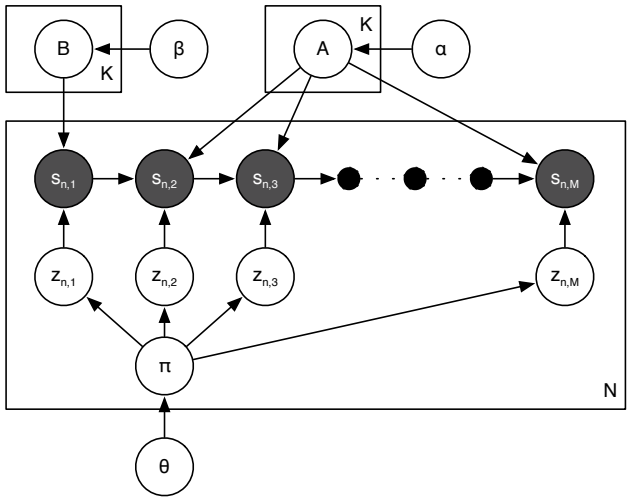


Figure 4: Finite Markov Model

Refer to Figure 4 for a graphical representation of Finite Markov Mixture Model. A $z_{n,m}$ is introduced as a latent variable for each element of a sequence $s_{n,m}$ to denote the cluster $s_{n,m}$ belongs to. Vector B and Matrix A now has rectangular boxes around it to denote that there are now K vectors of B and K matrices of A.

3.1 Generalization of Markov Model

Markov Mixture Model is a generalization of Markov Model. We show that using mixture models, we can divide the sequences S into K number of clusters. The sequences that have been allocated to a cluster k have high similarity within the cluster and low similarity to sequences in other clusters.

Here we state that there are K initial state probability matrices $B_k \in \mathbb{R}^{\mathbb{W}}$ and K transition state probability matrices $A_k \in \mathbb{R}^{\mathbb{W}\times\mathbb{W}}$. We let $z_{n,m} \in \{1, 2, \ldots, K\}$ denote the cluster that sequence element $s_{n,m}$ belongs. More formally, we can express the Markov Mixture Model as follows,

$$P(z_{n,m} = k|\pi_n) = \pi_{n,k} \tag{13}$$

$$\sum_{k=1}^{K} \pi_{n,k} = 1 \tag{14}$$

$$P(s_{n,1} = i|B, z_{n,1} = k) = b_{k,i} \tag{15}$$

$$P(s_{n,m+1} = j|s_{n,m} = i, A, z_{n,m+1} = k) = a_{k,i,j} \tag{16}$$

Equation 13 indicates the probability of sequence element $s_{n,m}$ belonging to cluster k. Equation 14 shows the consequence of probability, which sums to 1 in total. Equation 15 indicates which initial state probability vector we should use given that element $s_{n,m}$ belongs to cluster k. Equation 16 indicates which transition state probability matrix we should use given that sequence $s_{n,m}$ belongs to cluster k.

However, we do not know the value of each $z_{n,m}$, we do not know which cluster element $s_{n,m}$ belongs to. Hence, the model can be rewritten as follows,

$$\begin{aligned} &P(s_{n,1} = i|B, \pi_n) \\ &= \sum_{k=1}^{K} P(s_{n,1} = i, z_{n,1} = k|B, \pi_n) && (17) \\ &= \sum_{k=1}^{K} P(s_{n,1} = i|B, z_{n,1} = k)P(z_{n,1} = k|\pi_n) && (18) \\ &= \sum_{k=1}^{K} b_{k,i}\ \pi_{n,k} && (19) \end{aligned}$$

Equation 17 is the result of the law of total probability. Equation 18 is a consequence of Bayes Theorem.

And similarly,

$$P(s_{n,m+1} = j|s_{n,m} = i, A) = \sum_{k=1}^{K} a_{k,i,j}\pi_k$$

Suppose $K = 1$, then the Markov Mixture Model reduces back to the original Markov Model we discussed earlier.

3.2 Parameter Learning

Suppose we know the value of $z_{n,m}$ for each $s_{n,m}$, then the parameter update equations are similar to the original markov model. However, we do not know the value of $z_{n,m}$ for each $s_{n,m}$. Hence, we use Gibbs Sampling to infer the value of $z_{n,m}$ for each $s_{n,m}$. Before introducing Gibbs Sampling, we will first like to show that we approximate the

distribution of π_n using a symmetric dirichlet distribution,

$$(\pi_{n,1}, \ldots, \pi_{n,K}) \sim Dir(\theta) \tag{20}$$

$$P(z_{n,m}|\pi_1, \ldots, \pi_K) \propto \prod_{k=1}^{K} \pi_k^{\theta/K-1} \tag{21}$$

Note that there is only a single hyper-parameter θ instead of one hyperparameter for each dimension of the distribution. Similarly, the initial state vectors and transition matrices also follows symmetric dirichlet distributions.

$$A \sim Dir(\alpha)$$
$$B \sim Dir(\beta)$$

3.3 Inference using Gibbs Sampling

Gibbs Sampling is an approximate inference technique for sampling from joint density distributions. For example, suppose we want to sample from distribution of such form $P(x, y, z)$. We can do the following, assume we have initial values of x, y and z. We sample from $P(x|y, z)$ then use the new value of x to sample $P(y|x, z)$ then use new value of y to sample $P(z|x, y)$. Repeating such sampling forms an iterative markov chain and such sampling technique is known as Gibbs Sampling. Casella gives a very good introduction and explanation of why such technique works [2].

Gibbs Sampling is infact a statistical search algorithm which runs infinitely long to achieve a distribution that is near to the actual distribution. For ease of computation, we fix θ, β and α as $\mathbb{R}$ constant parameter values. There are ways to automatically decide these values, thus, fulfiling the automatic parameter learning. What we really want to infer here are the $z_{n,m}$ values. We want to form the Gibbs Sampler in this manner,

$$\begin{aligned}
&P(z_{1,1} = k|z_{1,2}, \ldots, z_{1,M}, \ldots, z_{N,M}) \\
&\quad \ldots \\
&P(z_{1,M} = k|z_{1,1}, \ldots, z_{1,M-1}, \ldots, z_{N,M}) \\
&P(z_{2,1} = k|z_{1,1}, \ldots, z_{1,M}, \ldots, z_{N,M}) \\
&\quad \ldots \\
&P(z_{2,M} = k|z_{1,1}, \ldots, z_{1,M}, \ldots, z_{N,M}) \\
&\quad \ldots \\
&P(z_{N,M} = k|z_{1,1}, \ldots, z_{1,M}, \ldots, z_{N,M})
\end{aligned}$$

Putting the Gibbs Sampling concept and writing it formally in the context of our sequence clustering here,

$$\begin{aligned}
&P(z_{n,1} = k|z_{n,-1}, S, \beta, \alpha, \theta) \\
&\propto \left[\frac{c_{k,i} + \frac{\beta}{W}}{\sum_i c_{k,i} - 1 + \beta}\right]\left[\frac{e_{n,k} + \frac{\theta}{K}}{\sum_k e_{n,k} - 1 + \theta}\right]
\end{aligned} \tag{22}$$

$$\begin{aligned}
&P(z_{n,m} = k|z_{n,-m}, S, \beta, \alpha, \theta) \\
&\propto \left[\frac{d_{k,i,j} + \frac{\alpha}{W}}{\sum_j d_{k,i,j} - 1 + \alpha}\right]\left[\frac{e_{n,k} + \frac{\theta}{K}}{\sum_k e_{n,k} - 1 + \theta}\right]
\end{aligned} \tag{23}$$

$z_{n,-m}$ represents the set of cluster assignments to all M elements of N sequences except for the m element of sequence n. Equation 23 (22) is the equation to use to decide which value of k to allocate to $s_{n,m}(s_{n,1})$ where $m \geq 2$ for a particular iteration.

3.4 Deciding the Clusters Probability

Finally, after sampling for the Z values, we will like to know what are the probabilities that a sequence n belongs to a cluster k. From Figure 4, the cluster probabilities are summarized by the variable π_n for each s_n. The distribution π_n may now be updated by the following posterior dirichlet distribution,

$$P(\pi_n|z_{n,1}, \ldots, z_{n,M}, \theta) \propto P(z_{n,1}, \ldots, z_{n,M}|\pi_n)P(\pi_n|\theta) \tag{24}$$

$$\propto \prod_{k=1}^{K} \pi_k^{e_{n,k}+\theta/K-1} \tag{25}$$

3.5 Optimizing the Hyper-Parameters Theta, Alpha and Beta

From the start of the Markov Model description, we have not mentioned how to decide the values of θ, α and β. We have always assumed that they are constants. Infact, they are not constants and their values should depend on the data we observed. This is the crucial point of nonparametric statistics. We shall show how to decide these values automatically. First we form the following equation which represents the likelihood of cluster assignments,

$$P(Z|\theta) = \prod_{n=1}^{N}\prod_{m=1}^{M} P(z_{n,m}|\theta) \tag{26}$$

$$= \prod_{n=1}^{N}\int \prod_{m=1}^{M} P(z_{n,m}, \pi_n|\theta)\ d\pi_n \tag{27}$$

$$= \prod_{n=1}^{N}\int \prod_{m=1}^{M} P(z_{n,m}|\pi_n)P(\pi_n|\theta)\ d\pi_n \tag{28}$$

$$\propto \prod_{n=1}^{N}\int \prod_{k=1}^{K} \pi_{n,k}^{e_{n,k}+\frac{\theta}{K}} \tag{29}$$

$$\propto \prod_{n=1}^{N}\left[\frac{\Gamma(\theta)}{\prod_{k=1}^{K}\Gamma(\theta/K)}\frac{\prod_{k=1}^{K}\Gamma(e_{n,k} + \frac{\theta}{K})}{\Gamma(\theta + \sum_{k=1}^{K} e_{n,k})}\right] \tag{30}$$

$$\propto \prod_{n=1}^{N}\left[\frac{\Gamma(\theta)}{\Gamma(\theta + \sum_{k=1}^{K} e_{n,k})}\prod_{k=1}^{K}\frac{\Gamma(e_{n,k} + \frac{\theta}{K})}{\Gamma(\theta/K)}\right] \tag{31}$$

The equation only has one unknown θ. The θ value should then maximize the equation. There are many ways to maximize such a function. Newton Raphson is one of them, but we shall show a Fixed Point Iteration Method [9]. The Fixed Point Iteration Method places a lower bound on the function that we want to maximize. According to Minka [9], these bounds are obtained from Milan Merkle's work [8]. We do not understand this work yet, but we hope to understand it one day.

Using the following lower bounds,

$$\frac{\Gamma(\theta)}{\Gamma\left(\theta+\sum_{k=1}^{K} e_{n,k}\right)} \geq d_n \; e^{\left(\theta^{old}-\theta\right)b_n} \quad (32)$$

$$b_n = \Psi\left(\theta^{old}+\sum_{k=1}^{K} e_{n,k}\right) - \Psi\left(\theta^{old}\right) \quad (33)$$

$$d_n = \frac{\Gamma\left(\theta^{old}\right)}{\Gamma\left(\theta^{old}+\sum_{k=1}^{K} e_{n,k}\right)} \quad (34)$$

$$\frac{\Gamma\left(e_{n,k}+\theta/K\right)}{\Gamma\left(\theta/K\right)} \geq c_{n,k}\left(\frac{\theta}{K}\right)^{a_{n,k}} \quad (35)$$

$$c_{n,k} = \frac{\Gamma\left(e_{n,k}+\frac{\theta^{old}}{K}\right)}{\Gamma\left(\frac{\theta^{old}}{K}\right)} \left(\frac{\theta^{old}}{K}\right)^{-a_{n,k}} \quad (36)$$

$$a_{n,k} = \left(\frac{\theta^{old}}{K}\right)\left(\Psi\left(e_{n,k}+\theta^{old}/K\right) - \Psi\left(\theta^{old}/K\right)\right) \quad (37)$$

Substitute the bounds above into Equation 31,

$$P(Z|\theta) \geq \prod_{n=1}^{N}\left[d_n \; e^{\left(\theta^{old}-\theta\right)b_n} \prod_{k=1}^{K}\left[c_{n,k} \;\; (\theta/K)^{a_{n,k}}\right]\right] \quad (38)$$

$$\log P(Z|\theta) \geq \sum_{n=1}^{N}\log\left[d_n \; e^{\left(\theta^{old}-\theta\right)b_n} \prod_{k=1}^{K}\left[c_{n,k} \;\; (\theta/K)^{a_{n,k}}\right]\right] \quad (39)$$

$$\geq \sum_{n=1}^{N}\left[\log d_n + b_n\left(\theta^{old}-\theta\right) + \sum_{k=1}^{K}\left[\log c_{n,k} + a_{n,k}\log\frac{\theta}{K}\right]\right] \quad (40)$$

Now to maximize the RHS, differentiate with respect to θ

$$\frac{d}{d\theta}\log P(Z|\theta) \geq \sum_{n=1}^{N}\left[-b_n + \sum_{k=1}^{K}\frac{a_{n,k}}{\theta}\right] \quad (41)$$

$$\geq \frac{\sum_{n=1}^{N}(\sum_{k=1}^{K} a_{n,k} - b_n\theta)}{\theta} \quad (42)$$

Equate the RHS to 0 and solve for θ.

$$\theta = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K} a_{n,k}}{\sum_{n=1}^{N} b_n} \quad (43)$$

$$= \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\frac{\theta^{old}}{K}\right)\left(\Psi(e_{n,k}+\theta^{old}/K) - \Psi(\theta^{old}/K)\right)}{\sum_{n=1}^{N}\left[\Psi(\theta^{old}+\sum_{k=1}^{K} e_{n,k}) - \Psi(\theta^{old})\right]} \quad (44)$$

$$= \frac{\theta^{old}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\Psi(e_{n,k}+\theta^{old}/K) - \Psi(\theta^{old}/K)\right)}{K\Big[-N\Psi(\theta^{old}) + \sum_{n=1}^{N}\Psi(\theta^{old}+\sum_{k=1}^{K} e_{n,k})\Big]} \quad (45)$$

$$= \frac{\theta^{old}\Big[-NK\Psi(\theta^{old}/K) + \sum_{n=1}^{N}\sum_{k=1}^{K}\Psi(e_{n,k}+\theta^{old}/K)\Big]}{K\Big[-N\Psi(\theta^{old}) + \sum_{n=1}^{N}\Psi(\theta^{old}+\sum_{k=1}^{K} e_{n,k})\Big]} \quad (46)$$

Run Equation 46 for some arbitrary iterations. we will skip the derivations for α and β, basically, it follows the same principle. The following shows the updates for α and β.

$$\beta = \frac{\beta^{old}\Big(-KW\Psi(\frac{\beta^{old}}{W}) + \sum_{k=1}^{K}\sum_{i=1}^{W}\Psi(c_{k,i}+\frac{\beta^{old}}{W})\Big)}{W\Big(-K\Psi(\beta^{old}) + \sum_{k=1}^{K}\Psi(\sum_{i=1}^{W} c_{k,i}+\beta^{old})\Big)} \quad (47)$$

$$\alpha = \frac{\alpha^{old}\Big(-KW^2\Psi(\frac{\alpha^{old}}{W}) + \sum_{k,i,j}^{K,W,W}\Psi(d_{k,i,j}+\frac{\alpha^{old}}{W})\Big)}{W\Big(-KW\Psi(\alpha^{old}) + \sum_{k,i}^{K,W}\Psi(\sum_{i=1}^{W} d_{k,i,j}+\alpha^{old})\Big)} \quad (48)$$

The appendix summarizes all of the clustering algorithm as discussed.

4. EXPERIMENTAL RESULTS

We use our clustering results for performing the Automatic Parameter Tuning as described in Section 1.5. We compare the TSP path results with [7] which uses the AGNES clustering algorithm. Table 4 shows the comparison of cluster assignment for training instances. Table 4 shows the comparison of cluster assignment for testing instances. The cluster assignments show little differences.

Table 4 shows the comparison of our results. The left most column represents the name of the Symmetric TSP instance from TSPLIB. The other two columns show the average differences between the best TSP path length and path length found by our algorithms. The results show that there is no significant difference between our model with Agnes.

Markov 0	Agnes 0	Markov 1	Agnes 1	Markov 2	Agnes 2
	a280	fl3795	fl3795	pr76	pr76
eil101	eil101	fl1400	fl1400	pr152	pr152
eil51	eil51	p654	p654	kroa100	kroa100
eil76	eil76	u2319	u2319	pr136	pr136
gil262	gil262	pr2392	pr2392	kroe100	kroe100
	pr226	d1655	d1655	krob100	krob100
	rat575	d2103	d2103	krob150	krob150
	rat783	u2152	u2152	kroc100	kroc100
st70	st70	rat783		kroa150	kroa150
		pcb1173	pcb1173	berlin52	berlin52
		fl1577	fl1577	kroa200	kroa200
		d657	d657	rd100	rd100
		nrw1379	nrw1379	lin105	lin105
		u1432	u1432	lin318	lin318
		vm1748	vm1748	u159	u159
		rat575		pr1002	
		rl1323	rl1323	vm1084	
		rl1889	rl1889	rat99	rat99
		pr299		d198	d198
		rl1304	rl1304	linhp318	linhp318
		d493	d493		krob200
		u1060	u1060		ch130
			vm1084		ch150
			pcb442		rat195
			pr1002		ts225
					pr299
					pr107

Table 2: Cluster Assignment for Training Instances

Markov 1	Agnes 1	Markov 2	Agnes 2
d1291	d1291	bier127	bier127
fl417	fl417	krod100	kroD100
pcb3038	pcb3038	pr124	pr124
u1817	u1817	pr144	pr144
u574	u574	pr264	pr264
u724	u724	pr439	pr439
tsp225		rd400	rd400
			tsp225

Table 3: Cluster Assignment for Testing Instances

Instance	Markov Mixture Model	AGNES
d1291	6.49	5.57
fl417	4.01	3.27
pcb3038	6.72	6.45
u1817	6.98	6.39
u574	5.67	5.70
u724	5.34	5.45
tsp225	3.34	4.00
bier127	2.34	2.42
krod100	2.34	3.12
pr124	1.42	1.11
pr144	0.66	1.04
pr264	7.38	8.98
pr439	5.27	4.48
rd400	4.36	4.55
Average	4.45	4.47

Table 4: Markov Mixture Model vs Agnes

5. DISCUSSION OF OTHER APPLICATIONS FOR DECISION SUPPORT

We have presented the Finite Markov Mixture Model in the previous sections. We have shown that Markov Mixture Models can be used for lustering sequences. However, Markov Mixture Models (MMM) have applications that extends beyond clustering. MMM is infact a more fine-grained generalization of Markov Models. Using MMM, we can observe sequences of events that occur in real world systems to predict for,

1. The occurence of future events.
2. When will certain state events occur.

By knowing what will happen in the future, will give us knowledge about making decisions in the present. Combining this work and other research of Artificial Intelligence, we are now one step closer to making real world Analytical Systems.

6. ACKNOWLEDGMENTS

The author(s) will like to thank Prof Hoong Chuin Lau and Lindawati for providing the ideas and help to making this project possible.

7. REFERENCES

[1] B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Oper. Res.*, 54(1):99–114, 2006.

[2] G. Casella and E. I. George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[3] F. X. H. C. Lau and S. Halim. Enhancing the speed and accuracy of automated parameter tuning in heuristic design. *8th Metaheuristics International Conference*, 2009.

[4] H. Hoos and T. Stutzle. *Stochastic Local Search: Foundation and Application*. Morgan Kaufmann, 2004.

[5] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[6] Y. Liang and A. Kelemen. Bayesian finite markov mixture model for temporal multi tissue polygenic patterns. *Biometrical*, 51:56–69, 2009.

[7] D. L. Lindawati, H. C. Lau. Generic instance-based automated parameter tuning via search trajectory similarity clustering. *To appear European Conference on Artificial Intelligence*, 2010.

[8] M. Merkle. Conditions for convexity of a derivative and applications to the gamma and digamma function. *Ser. MATH. INFORM*, 16:13–20, 2001.

[9] T. P. Minka. Estimating a dirichlet distribution. 2003.

[10] H. Ramalhinho-Lourenco, O. C. Martin, and T. Stutzle. Iterated local search. Economics Working Papers 513, Department of Economics and Business, Universitat Pompeu Fabra, Nov. 2000.

[11] P. Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. MIT Press, 1997.

APPENDIX

A. OVERVIEW OF THE CLUSTERING ALGORITHM

Algorithm 1
Input: N TSP Training instances $\{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$
Input: K number of clusters
Output: Cluster membership $\{\pi_1, \ldots, \pi_N\}$

$\{s_1, \ldots, s_N\} \leftarrow ILS(\{\mathcal{G}_1, \ldots, \mathcal{G}_N\})$
Each sequence s_n is an ordered set $\{s_{n,1}, \ldots, s_{n,M}\}$.
Each element $s_{n,m}$ in sequence s_n has states as shown in Table 1.
Initialize $z[N][M], c[K][7], d[K][7][7], e[N][K]$
Initialize hyperparameters θ, α, β
for $n \leftarrow 1$ to N **do**
  $z[n][1] \leftarrow$ Randomly Assign 1 to K
  $e[n][z[n][1]] + +$
  $c[z[n][1]][s[n][1]] + +$
  $M \leftarrow$ length(s[n])
  for $m \leftarrow 2$ to M **do**
    $z[n][m] \leftarrow$ Randomly Assign 1 to K
    $e[n][z[n][m]] + +$
    $d[z[n][m]][s[n][m-1]][s[n][m]] + +$
  end for
end for
while not converge **do**
  for $n \leftarrow 1$ to N **do**
    $e[n][z[n][1]] - -$
    $c[z[n][1]][s[n][1]] - -$
    $z[n][1] \leftarrow$ Sample using Equation 22
    $e[n][z[n][1]] + +$
    $c[z[n][1]][s[n][1]] + +$
    $M \leftarrow$ length(s[n])
    for $m \leftarrow 2$ to M **do**
      $e[n][z[n][m]] - -$
      $d[z[n][m]][s[n][m-1]][s[n][m]] - -$
      $z[n][1] \leftarrow$ Sample using Equation 23
      $e[n][z[n][1]] + +$
      $d[z[n][m]][s[n][m-1]][s[n][m]] + +$
    end for
  end for
  $\theta \leftarrow$ Update using Equation 46
  $\beta \leftarrow$ Update using Equation 47
  $\alpha \leftarrow$ Update using Equation 48
end while
Finally, evaluate π_n using Equation 25

Algorithm 2
Input: N TSP Testing instances $\{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$
Input: $c[K][7], d[K][7][7]$ from previous algorithm
Output: Cluster membership $\{\pi_1, \ldots, \pi_N\}$

$\{s_1, \ldots, s_N\} \leftarrow ILS(\{\mathcal{G}_1, \ldots, \mathcal{G}_N\})$
Initialize $z[N][M], e[N][K]$
for $n \leftarrow 1$ to N **do**
  $M \leftarrow$ length(s[n])
  for $m \leftarrow 1$ to M **do**
    $z[n][m] \leftarrow$ Randomly Assign 1 to K
    $e[n][z[n][m]] + +$
  end for
end for
while not converge **do**
  for $n \leftarrow 1$ to N **do**
    $e[n][z[n][1]] - -$
    $z[n][1] \leftarrow$ Sample using Equation 22
    $e[n][z[n][1]] + +$
    $M \leftarrow$ length(s[n])
    for $m \leftarrow 2$ to M **do**
      $e[n][z[n][m]] - -$
      $z[n][1] \leftarrow$ Sample using Equation 23
      $e[n][z[n][1]] + +$
    end for
  end for
end while
Finally, evaluate π_n using Equation 25