A Swarm-Based Approach to Learning Phase-Type Distributions for Continuous Time Bayesian Networks

Logan Perreault, Monica Thornton, Rollie Goodman, and John W. Sheppard
Montana State University,
Bozeman, MT 59717 USA
{logan.perreault, monica.thornton, rollie.goodman, john.sheppard}@cs.montana.edu

***Abstract*—The use of phase-type distributions is an established method for extending the representational power of continuous time Bayesian networks beyond exponentially-distributed state transitions. In this paper, we propose a method for learning phase-type distributions from known parametric distributions. We find that by using particle swarm optimization to minimize a modified KL-divergence value, we are able to efficiently obtain good phase-type approximations for a variety of parametric distributions. Our experiments show that particle swarm optimization outperforms genetic algorithms and hill climbing with simulated annealing. In addition, we investigate the trade-off between accuracy and complexity with respect to the number of phases in the phase-type distribution. Finally, we propose and evaluate an extension that uses informed starting locations during optimization, which we found to improve convergence rates when compared to random initialization.**

I. Introduction

It is necessary in many domains to model a system as it evolves through time. This can be accomplished by discretizing time into uniform slices. However, this approach requires the selection of a time granularity, and this can be inefficient when there is no natural choice of granularity in the system under study. An alternative approach is to model time as continuous, which explicitly represents temporal dynamics while eliminating the need for a uniform time granularity.

Continuous time Bayesian networks (CTBNs) provide a framework for modeling time directly, and numerous extensions have expanded the representational power of the model. In their original form, CTBNs can only model systems with exponentially distributed duration times. To address this, a method has been recently developed to allow CTBNs to model systems with durations that occur according to a variety of distributions. In this paper, we apply particle swarm optimization to parameterize phase-type distributions, the use of which allows the CTBN framework to support non-exponential duration times.

A. Continuous Time Bayesian Networks

A Markov process is a memoryless stochastic process, meaning that future states of the process are independent of previous states given the current state. A continuous time Markov process (CTMP) describes the evolution of a process in continuous time over a finite set of discrete states. A CTMP is represented as an initial distribution over the states, and a square transition intensity matrix Q with dimensions equal to the number of states in the process. An element q_{ij} in row i, column j of the matrix indicates the intensity with which the process transitions from state i to state j. More formally, the time it takes to transition between the states i and j is drawn from an exponential distribution with a rate that corresponds to the entry q_{ij}. The diagonal entries in Q are defined to be the negative sum of the remaining entries in the row, which ensures that every row in the matrix sums to zero. Any row with a nonzero entry represents a state that will eventually transition to another, and is referred to as a transient state. If a row contains all zeros, there is no way to transition to another state and it is therefore considered an absorbing state. A Markov process will eventually transition into an absorbing state if one exists, and will remain in that state permanently.

A CTBN is a factored representation of a Markov process [1]. This representation has two components: an initial distribution over the variables in the model, and a set of conditional intensity matrices (CIMs) for each variable. The factorization is achieved by taking advantage of conditional independencies, which are encoded using a directed graph, the nodes of which represent the variables in the process. Each CIM is an intensity matrix conditioned on an instantiation of the node's parents. The CTBN formulation provides a framework that can model time directly in a way that other temporal models, such as dynamic Bayesian networks, cannot.

Although CTBNs have been successfully used to model system change over time, the framework does have practical limitations. In particular, the computational efficiency achieved by CTBNs is rooted in their exploitation of the Markov property, which is reliant on the use of exponential distributions. As a consequence, CTBNs in their original form can only describe processes with exponentially distributed duration times. This restriction can be very limiting when it comes to modeling real-world domains. The desire to retain the computational advantages afforded by CTBNs while simultaneously expanding their scope to describe more complex distributions has motivated the use of phase-type distributions as an extension to the CTBN framework.

Phase-type distributions, represented by the time until absorption in a Markov process, can be used to closely approximate any positive parametric distribution. This versatility, combined with their adherence to the Markov property, makes

978-1-4799-7560-0/15/$31

phase-type distributions especially valuable in the context of CTBNs. Recent work has shown that phase-type distributions can be used to extend the expressive power of CTBNs by transforming each state within the CIM into multiple phases, and using the absorbing phase to define an exit distribution to another state in the system [2]. This allows the CTBN to retain the Markov property and the related efficiency gains, while also providing a more expressive model that can approximate a greater number of distributions. A more formal treatment of phase-type distributions can be found in Section II-C of this paper.

B. Motivation

To date, work on extending CTBNs through the use of phase-type distributions has focused on the problem of approximating a distribution that describes available data. This has been achieved by applying expectation-maximization algorithms to learn a model that optimizes the log-likelihood of the data. While this approach works well in the presence of abundant data, in many practical applications there is inadequate data to describe a distribution sufficiently. Fitting a distribution to data requires a minimum amount of data points to measure goodness of fit [3]. In cases where there is insufficient data, a viable alternative is to rely on domain knowledge to indicate transition rates with a well-defined probability distribution. These distributions are known to describe the behavior of a system accurately, and as such we will refer to these probability distributions as "true" distributions.

A true distribution, unlike an unknown distribution described by available data, is specified entirely by its parameters. Using true distributions as a means to describe system behavior is an established practice when performing tasks such as failure rate analysis [4]. Learning phase-type distributions from true distributions provides information about transition times for a variable while avoiding the need for significant amounts of data.

To this end, we propose casting the learning process as an optimization problem that seeks to minimize the KL-divergence of a learned phase-type distribution from the true distribution. This is equivalent to maximizing the closeness of the approximation. We chose to use particle swarm optimization (PSO), a well-known swarm intelligence algorithm, to solve this optimization problem. PSO has been successful in related areas and has many useful extensions; therefore, we hypothesize that it is well-suited to our application.

II. Parametric Distributions

Our work is primarily motivated by a desire to model system failures. For this reason, we focus our discussion on a subset of distributions that are commonly used to model time-to-failure (TTF); specifically, the Weibull and lognormal distributions [5]. Exponential distributions are also frequently used to model failure distributions, but CTBNs are already naturally suited to model these distributions. We note that although we limit the scope of our discussion to Weibull and lognormal distributions, the framework we propose allows for the approximation of any positive true distribution.

A. Weibull Distribution

Weibull is a flexible distribution parameterized by a rate parameter λ and a shape parameter k [6]. Consider the case where the input t to the distribution is interpreted as the TTF. When $k < 1$, the Weibull distribution represents a decreasing failure rate, and an increasing failure rate when $k > 1$. In the special case where $k = 1$, the Weibull distribution reduces to an exponential distribution with a rate of λ. When t is positive, the probability density function (PDF) for the Weibull distribution is defined as follows:

$$f(t;\lambda,k) = \frac{k}{\lambda}\Big(\frac{t}{\lambda}\Big)^{k-1} \exp\Big(-\Big(\frac{t}{\lambda}\Big)^{k}\Big) \ .$$

Since the failure rate for a Weibull distribution can be increasing, decreasing, or constant, it can be used to model the "infant mortality" stage (where $k < 1$), the "useful life" stage (where $k = 1$), and the "end of life" stage (where $k > 1$) of an object. When spliced together in this order, these three piecewise segments form what is often called the "bathtub" curve. This is considered an appropriate model for the failure rate of many objects, as it reflects a higher rate of failure surrounding the birth and death of an object, and a smaller constant rate of failure during the rest of the object's lifespan. Additionally, it has been suggested that this composite approach may prove to be a more realistic model for TTF than monotone failure rate models [7]. The plot for this distribution using several different parameterizations is shown in Figure 1.

B. Lognormal Distribution

Another frequently used TTF curve is the lognormal distribution, which indicates that the log of a random variable follows a normal distribution. Due to its dependence on the normal distribution, the only necessary parameters for the lognormal distribution are the mean parameter μ and the standard deviation parameter σ. The lognormal distribution is suited for instances where failure occurs due to an accumulation of causes that have a multiplicative effect, a phenomenon known as multiplicative degradation [8]. When the input t is positive, the PDF for the lognormal distribution is as follows:

$$f(t;\mu,\sigma) = \frac{1}{t\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln t - \mu)^2}{2\sigma^2}\right) \ .$$

Figure 2 provides a plot of various parameterizations for this distribution.

C. Phase-Type Distributions

Phase-type distributions are a semi-parametric class of distributions that use exponential distributions as a means to approximate general positive distributions, including (but not limited to) both Weibull and lognormal. Formally, phase-type distributions represent the time until absorption in a Markov process with n transient states and one absorbing state [9]. The transient states in the Markov process are also referred to as the phases of the phase-type distribution. A variable will move through these phases according to the exponential distributions defined for each phase by the rates in the Markov process, until the variable eventually reaches the absorbing state. The phase-type distribution is defined as the distribution of the entire time it takes the process to move through these phases.

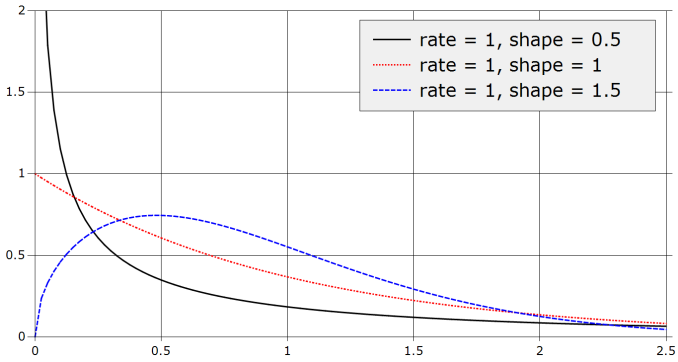


Fig. 1: PDFs for various parameterizations of Weibull distributions.

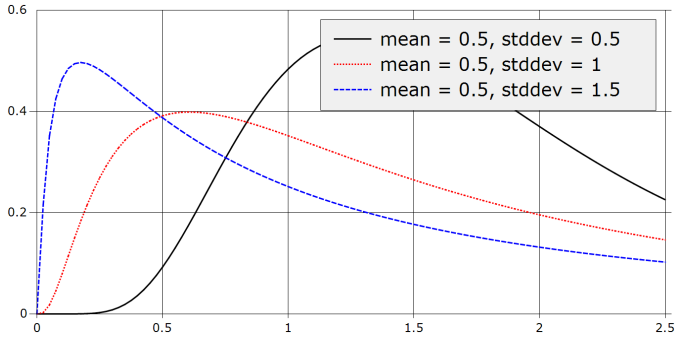


Fig. 2: PDFs for various parameterizations of lognormal distributions.

Since a phase-type distribution depends only on a Markov process, its parameters are specified fully by the initial distribution and transition intensity matrix for the Markov process. It is generally assumed that the probability of starting in the absorbing state is zero, and in many cases it is further restricted to ensure that the process starts in the first transient phase. The movement of a variable through the transient phases of the Markov process can be directed in a variety of ways. In the most general case, all transient phases are capable of transitioning to any other phase, including the absorbing state. Specific classes of phase-type distributions restrict the movement of a variable between the transient phase. Several types of restricted phase-type distributions are discussed in the review of related literature.

The PDF for a phase-type distribution is given as:

$$f(t) = \alpha \exp(\mathbf{S}t)\mathbf{S^0} \tag{1}$$

where α is a vector corresponding to the probability of starting in each phase, $\mathbf{S^0}$ is a vector of intensities for transitioning to the absorbing state from each of the other phases, and $\mathbf{S}$ is a square matrix of intensities for transitioning between non-absorbing phases. The matrix exponential operation, $\exp(\mathbf{S})$, is defined by the Taylor series as shown.

$$\exp(\mathbf{S}) = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{S}^k \tag{2}$$

Calculating the matrix exponential is generally intractable. Fortunately, there exist a variety of methods for calculating an approximate matrix exponential [10]. We chose to utilize the most commonly used approximation, known as the scaling and squaring method [11].

The idea behind scaling and squaring is to scale the matrix by a power of two to reduce the norm to order one, and then compute a Padé approximant[1] for the matrix exponential on the scaled matrix. Repeatedly squaring the resulting matrix undoes the scaling. By using the scaling and squaring method to calculate the matrix exponential, approximating the PDF of a phase-type distribution becomes tractable. The ability to approximate this PDF becomes important when evaluating the quality of a phase-type approximation for a true distribution.

A phase-type distribution can be embedded directly into a CIM for a CTBN without significantly changing the usage of the CTBN [13]. A single row in a CIM can be replaced by entries calculated using the phase-type distribution and the original multinomial distribution defined by the row. When performing inference over the CTBN, a node is interpreted as being in a particular state if it is currently in any of the phases that have been inserted for the state. The underlying mechanics remain unchanged, since phase-type distributions are built using exponential distributions, which adhere to the Markov assumption.

III. Related Work

This paper fills a gap in the literature, as PSO and similar population-based methods have not yet been applied to CTBNs. However, related areas contain research that validates the choice of PSO for our optimization task. For instance, PSO was used to maximize a symmetric version of KL-divergence to assist in a feature transformation procedure for a speaker-verification application [14]. This work was of interest to us because we also use PSO to optimize KL-divergence. In this case, the goal was to maximize a KL-divergence approximation specific to Gaussian mixture models, a common model for speaker-verification systems. The PSO-based method was found to reduce the overall error rate of verification as compared to the baseline method.

PSO has also been applied to learning Bayesian networks (BNs) for fault-diagnosis applications. This is related to our work in that CTBNs share many similarities with BNs and can be used for similar applications. Sahin *et al.* used a distributed PSO algorithm for an application in learning BN structure from sensor data [15]. Rather than optimizing a distance measure, in this case PSO optimized a network-evaluation score that measures the optimality of candidate BN structures.

Fortier *et al.* used a multi-population variant of PSO, called Overlapping Swarm Intelligence (OSI), to estimate parameters in BNs [16]. OSI is a multi-swarm PSO algorithm, where swarms are assigned to overlapping portions of the larger search space and inter-swarm communication maintains a global solution. OSI significantly outperformed the competing approaches, including traditional PSO, a genetic algorithm, and expectation-maximization (EM). It is worth noting that while the results of the previously mentioned studies are promising,

[1] An [n,m] Padé approximant is a rational function consisting of a polynomial of degree m divided by a polynomial of degree n, which is useful for providing an approximation of the power series of another function [12].

a BN cannot model continuous time; therefore, a CTBN is better suited to many real-world applications.

The use of phase-type distributions for CTBNs was first proposed by Nodelman and Horvitz [13]. They demonstrated that phase-type distributions can be inserted as subsystems in a CTBN without altering the underlying mechanics of the model. Their work was restricted to Erlang distributions, a subclass of phase-type distributions where loops are not permitted, and each phase is required to have the same rate parameter and must be visited in order before transitioning to the absorbing state. The use of an Erlang distribution was shown to improve the performance of the CTBN model when the underlying distribution was non-exponential. The primary contribution made by the authors is the notion that phase-type distributions can be inserted into a CTBN without changing the framework as a whole. The specific details of how to parameterize the phase-type distributions, however, were omitted.

The work by Nodelman and Horvitz was later extended from Erlang distributions to Erlang-Coxian distributions, which are another more expressive subclass of phase-type distributions [2]. A Coxian distribution is similar to an Erlang distribution in that it does not permit cycles in the phases. In contrast, however, a Coxian distribution may be uniquely parameterized at each phase, and any of the phases may transition to the absorbing state. An Erlang-Coxian distribution combines these two ideas by forcing sequential progression through the Erlang phases until the Coxian phases are reached, at which point the phase may either transition to the next phase, or go to the absorbing state directly.

The addition of the Coxian phases increases the expressiveness of the model, but at the cost of added complexity through an increase in the required number of parameters. Instead of requiring a single rate parameter λ, two additional parameters are required for each phase in the Coxian distribution. This additional complexity can be managed by restricting the Coxian distribution to only two phases. Gopalratnam *et al.* propose a method for learning these parameters from data based on EM. Unlike the method we propose here, their technique attempts to parameterize the distribution such that the log-likelihood between the data and the model is maximized.

After demonstrating the utility of Erlang distributions as subsystems in CTBNs, in subsequent work Nodelman *et al.* discuss a method for performing EM and structural EM (SEM) to learn the parameters and structure of a CTBN model from partially observed data [17]. This EM algorithm further relaxes the restrictions on which subclasses of phase-type distributions are applicable, such that any phase-type distribution can be used. This allows phases to occur within a loop, significantly improving the expressiveness of the model. The experiments again showed a marked improvement over a CTBN model that did not use phase-type distributions. Nodelman *et al.* demonstrate the utility of this approach, but we explore a different angle with our contributions. While Nodelman learns phase-type distributions from data, we learn the parameters for a phase-type distribution to fit a known distribution.

IV. Approach

The primary goal of our research is to provide a method for learning a phase-type distribution that accurately approximates a given true distribution. To achieve this, we first cast the parameterization of the approximation distribution as an optimization problem. We then solve the optimization problem such that the result is a parameterization of a phase-type distribution that accurately approximates the specified true distribution.

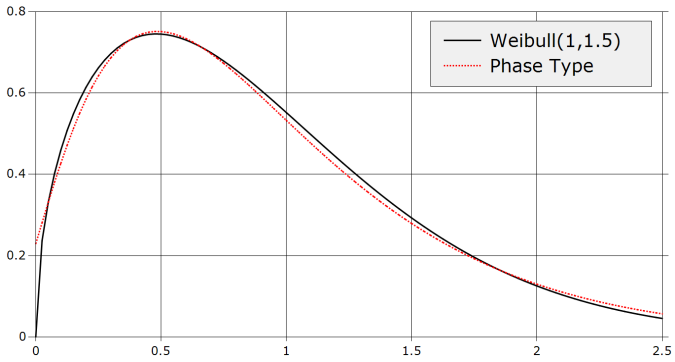


Fig. 3: Example of a phase-type distribution fitted to a Weibull distribution with a rate of 1.0 and a shape of 1.5.

As an example, consider a Weibull distribution with a rate parameter of 1.0 and a shape parameter of 1.5. The learned initial distribution and transition matrix for a phase-type distribution that approximates this Weibull distribution is as follows:

$$P(X) = (0.98 \quad 0.02 \quad 0.00)\,,$$

$$Q = \begin{pmatrix} -2.09 & 1.98 & 0.11 \\ 0.00 & -1.99 & 1.99 \\ 0.00 & 0.00 & 0.00 \end{pmatrix}.$$

A plot of the PDFs for both the Weibull distribution and the approximating phase-type distribution is shown in Figure 3. For reference, the goodness of fit for this approximation is quantified by a KL-divergence value of 0.0361.

We chose PSO as our optimization method due to its success in related applications. Additionally, it is an anytime algorithm, which gives the user the option to accept a suboptimal solution prior to convergence. We measured the performance of PSO by comparing it against two other well-established optimization algorithms.

A. Kullback-Leibler Divergence

In this work, we utilize the Kullback-Leibler divergence measure, often referred to simply as KL-divergence. KL-divergence serves as a measure of the information lost when approximating a true distribution with an approximating distribution, and is equal to zero if the two distributions are identical. Using this principle, we are able to construct an accurate approximation by choosing phase-type parameters that minimize the KL-divergence of the phase-type distribution from the true distribution.

The KL-divergence of distribution Q from distribution P is denoted as $D_{KL}(P||Q)$. For the case when P and Q are continuous, KL-divergence is defined by Equation 3. In practice, a discrete approximation can be used that evaluates the PDF for each distribution at specified intervals. In this

work, we use the equation specified by Equation 4. Note that in addition to evaluating the distributions at discrete intervals, we restrict the upper bound of the summation to be a finite value less than infinity. We have manually specified an upper bound of 2.5 since we have deemed it the area of interest for the distribution. An automated approach could be taken that chooses the upper bound based on the percentage of the distribution covered, as determined by the cumulative distribution function (CDF). Furthermore, we have introduced an absolute value to the log term, which is a modification to the standard KL-divergence equation. This is done to avoid situations where underestimation in one section of the approximate PDF might mask overestimation in another, which may occur due to the fact that the log term can be positive or negative.

$$D_{KL}(P||Q) = \int_0^\infty P(t)\, \log \frac{P(t)}{Q(t)}\, \mathrm{d}t \tag{3}$$

$$\approx D_{KL}(P||Q) = \sum_{i=1}^{n} P(i)\, \left| \log \frac{P(i)}{Q(i)} \right| \tag{4}$$

The calculation in 4 requires knowledge of the PDFs of both distributions. The PDF for the true distribution is unique to each distribution, but it is generally a trivial calculation. We use an approximation of the PDF for a phase-type distribution, along with the PDF for the true distribution, to compute the KL-divergence as described in Equation 4.

B. Optimization

Selecting parameters for the phase-type approximation of a true distribution is an optimization problem that seeks to minimize KL-divergence. PSO has demonstrated utility in solving these types of problems, and for a baseline comparison we also implemented a genetic algorithm (GA) and a simple hill-climbing algorithm that uses simulated annealing.

1) Particle Swarm Optimization: PSO begins by initializing a population of particles, each of which has a position in the search space that represents a possible candidate solution [18]. The quality of each particle's position can be evaluated using a fitness function that is problem-specific. Particles move through the search space as defined by their velocity, which is updated during each iteration of the algorithm according to the following velocity update equation:

$$v_i = \omega v_i + U(0, \phi_1) \otimes (p_i - x_i) + U(0, \phi_2) \otimes (p_g - x_i).$$

Here, v_i is the velocity for particle i, x_i is the position of particle i, p_i is the best position seen by particle i, and p_g is the best position seen by any particle in the population. For our purposes, KL-divergence is the fitness function, and the "best" solution is defined to be the solution with the lowest KL-divergence value. The first term is known as *momentum*, which pulls the particle in the direction it was previously going. The second term is the *cognitive component*, which draws the particle toward the best solution it has ever found, and the third is the *social component*, which draws the particle toward the best solution any particle in the swarm has ever found. The parameters ϕ_1, ϕ_2, and ω are user-defined constants which are manually tuned to control the degree to which each term influences the particle's movement.

The entire population is collectively referred to as a swarm, and the behavior of a particle is intended to mimic the social behavior of animals, such as flocking birds or schooling fish. The effect of the velocity update equation is that an individual particle is drawn toward three locations based on the momentum, the cognitive component, and the social component. This social component achieves the desired flocking behavior.

2) Genetic Algorithm: Genetic algorithms, as their name suggests, are an attempt to bring the advantages of Darwinian evolution to optimization algorithms [19]. The idea behind a genetic algorithm is to start with a population of randomly generated solutions, which are in this case assignments of values to the parameters of a phase-type distribution. Each solution in the population is called an *individual* or *chromosome*, and the fitness for these solutions is calculated as their KL-divergence from the true distribution. Iteratively, a new "offspring" population is created from the existing population as follows.

First, two parent chromosomes are chosen such that assignments with lower KL-divergence are more likely to be selected. In our genetic algorithm implementation, we used tournament selection. This compares a small pool of candidate parents uniformly selected from the population and chooses a parent from this pool using a probability distribution weighted by fitness. The parent is then returned to the population and another parent is selected in the same manner; thus, it is possible to have the same chromosome as both parents.

Once the parents have been selected, crossover takes place. We utilized multi-point crossover, which involves randomly selecting sections of parameters and swapping them between the parents to create two offspring chromosomes. For our application we found that five-point crossover, which involves swapping five sections, gave the best results. The resulting offspring are then mutated, a process that randomly changes the values of some of their parameters by a small amount.

This process repeats until a desired number, n, of new offspring have been created, at which point the generation is completed and is used to replace n individuals from the old population. The algorithm is repeated for a specified number of generations, after which the fittest individual in the population is returned as the solution.

3) Hill-Climbing with Simulated Annealing: Hill-climbing using simulated annealing is a numerical analogue to the process of slowly cooling metals so that they crystallize at their minimum energy state, to improve upon the basic hill-climbing search [20].

In the original hill-climbing algorithm, the candidate solution, which for our purposes is a parameterization of the model, is initialized at a random state in the search space. Then, at each iteration, a random neighbor state is considered. If the neighboring state is found to have a better fitness value than the current state, the neighbor is accepted and becomes the current state. For our implementation, a better fitness value is defined as a lower KL-divergence. Simulated annealing introduces an extra step at each iteration to avoid becoming stuck in local optima. Once a neighbor state is selected, if the neighbor is worse, it is accepted or rejected based on the acceptance probability:

$$P(accept) = \exp\left(\frac{\mathrm{energy}(current) - \mathrm{energy}(neighbor)}{kT}\right)$$

where the energy of a given parameterization is its KL-divergence from the true distribution, k is the Boltzmann constant, and T is a value known as the *temperature*, which is initialized to some positive number and is slightly decreased at each iteration. A better neighbor state will still always be accepted, but now a worse solution may also be accepted based both on how much worse it is and on the temperature at that iteration.

The gradual lowering of the temperature parameter produces the desired annealing effect: the likelihood of accepting a worse solution is initially high but decreases as a function of time. This enables the algorithm to avoid becoming stuck in local optima early in the search process while still converging on a close-to-optimal solution in the later iterations of the search.

V. Experiments

In addition to comparing optimization algorithms, we explored the effects of several other factors on the ability to learn phase-type approximations. Specifically, we were also interested in how much an increase in the number of phases in the model added to its expressive power. Additionally, we proposed and evaluated alternatives to random initialization of solutions during optimization. The relevant experiments are detailed in the remainder of this section.

A. Optimization Methods

We ran the PSO, genetic algorithm, and hill-climbing with simulated annealing (HC) optimization algorithms on various parameterizations of Weibull and lognormal distributions. For the Weibull distribution we varied both the rate and shape parameters from 0.5 to 2.0 by increments of 0.1 for a total of 225 instances. The same values were used with the lognormal distribution for the mean and standard deviation parameters.

For this initial experiment, we fixed the size of the learned distribution to three phases. We used general phase-type distributions, which have no structural restrictions. The intensity matrix and initial distribution for each phase-type distribution were serialized so that they conformed to the optimization algorithm frameworks. This was done by extracting all non-diagonal entries from the intensity matrix excluding the last row, since the diagonal entries for each row of the intensity matrices can be computed as the negative sum of the rest of the row, and the last row was always set to 0 since the corresponding state is absorbing. We also included all but the last value of the initial distribution in the serialization, since the last value is the complement of the remaining values, such that the initial distribution sums to one. The result is a vector of size $\Theta(n^2)$, where n is the number of phases. We bounded the valid search space for the optimization such that the initial distribution values were smaller than one, while the entries for the intensity matrix are bounded by some positive user-specified value which we set to 2.00 for all experiments. The deserialization process reverses the described procedure to produce a phase-type distribution that can be used to evaluate candidate solutions for the optimization algorithms.

Algorithms were compared using the KL-divergence between the final learned phase-type distribution and the true distribution. The parameters for each algorithm were manually tuned. Hill-climbing used an initial temperature of 100 and decreased this value by a factor of 0.05 at each iteration. For the genetic algorithm, we used 100 individuals, five-point crossover, and two-parent tournament selection with a 75% chance of choosing the fittest parent. PSO used five particles, the behavior of which was dictated by a velocity update equation with a momentum of 0.9, a personal learning rate of 1.0, and a social learning rate of 1.5. Each of the algorithms were run for 1000 iterations, which in all cases appeared to be sufficient for convergence.

TABLE I: Comparison of Optimization Algorithms

	PSO	GA	HC
Weibull	**0.0498**	0.0815	0.2996
Lognormal	**0.0154**	0.0394	0.0888

The mean performances for each algorithm measured in KL-divergence from the target distribution are summarized in Table I . Algorithms were compared with the Wilcoxon signed-rank test with a confidence level of 0.95, a nonparametric test chosen because the datapoints originate from different distributions. Based on our statistical analysis, PSO significantly outperforms the GA, which in turn significantly outperforms HC. For this reason, we focus the rest of our experiments on how well PSO performs under varying conditions and omit any further results for GA and HC.

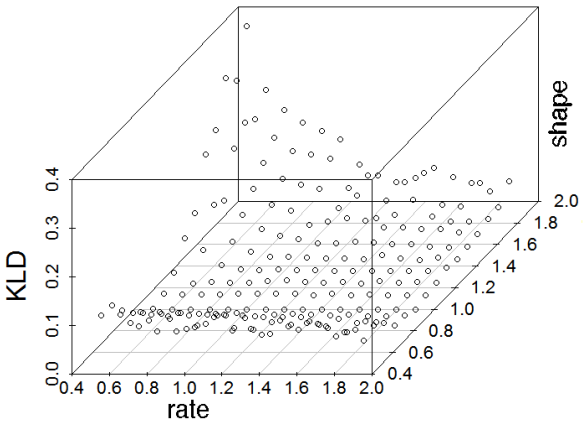
We also investigated PSO's ability to approximate distributions over different regions of the parameter space. As specified above, we used the 225 parameterizations for Weibull and lognormal and plotted the KL-divergence values obtained using PSO in Figure 4. Figure 4a illustrates that relatively low KL-divergence values are obtained for the majority of the search space, with the exception of those cases when the rate is low and the shape is high. Similarly, we find from Figure 4b that most parameterizations of the lognormal distribution can be approximated well, but performance degrades when both the mean and standard deviation are low.

B. Number of Phases

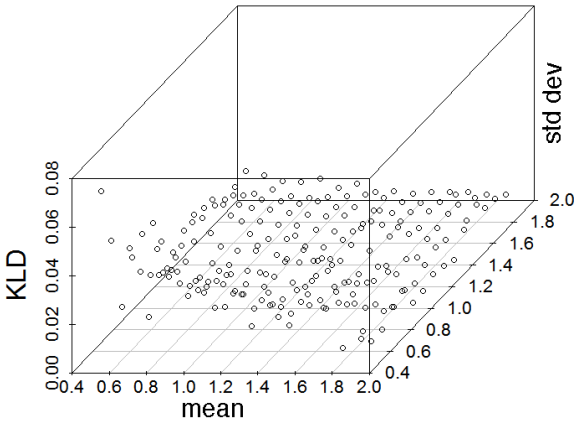
The next experiment investigated the effect of varying the number of phases in the phase-type distribution. Phase-type distributions with more phases have more representational power, and therefore allow for a more accurate approximation. However, more phases implies more parameters in the resulting CTBN, and therefore increased model complexity.

For this experiment, we used five representative parameterizations each for the Weibull and lognormal distributions. For the Weibull distribution, these values were $(0.8, 1.7), (1.0, 1.5), (1.3, 1.7), (0.7, 0.7)$ and $(1.0, 0.5)$, where the first value in each pair is the rate and the second value of the pair corresponds to the shape. In the case of the lognormal distribution, we used $(0.8, 1.2), (1.0, 1.2), (1.0, 1.0), (1.2, 1.0)$ and $(0.95, 1.0)$, where the first value of each pair is the mean and the second is the standard deviation.

For each of these ten true distributions, we used PSO to learn phase-type distributions with varying numbers of phases. Specifically, we started with a single phase (which is equivalent to the exponential distribution) and increased incrementally to



(a) Approximating Weibull



(b) Approximating lognormal

Fig. 4: KL-divergence values for approximating parameterizations of Weibull and lognormal distributions using PSO.

ten phases. Our analysis of these results consisted of a series of Wilcoxon signed-rank tests with a confidence level of 0.95. The single-phase distribution was significantly outperformed by every other case. We also found that the use of only two phases was significantly worse than using three, four, five, and six phases. In addition, using six phases was significantly better than using seven and eight, as well as one and two.

C. Informed Initialization

Our final experiment tested an extension to our proposed algorithm, which we call informed initialization. The idea is that rather than initializing particles randomly, we can use a more intelligent starting solution. This is accomplished by first approximating a variety of distributions using random initialization and saving the resulting parameters for the phase-type distributions. When learning a new distribution, the algorithm

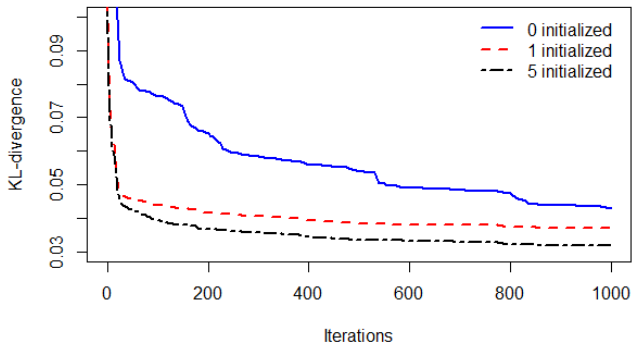


Fig. 5: Effect of informed initialization on KL-divergence.

can then initialize particles using a similar cached solution by calculating the sum of the differences between parameters and sorting the list.

For this experiment, the set of saved solutions was generated using the experiments run in Section V-A, resulting in 225 potential starting positions for each distribution. For the true distributions, we used the ten distributions discussed in Section V-B and the parameters for each were perturbed by $+0.05$ so that they could not be found exactly in the set of saved solutions. We varied the number of particles in the swarm that were initialized using a saved solution from zero (equivalent to random initialization) to all five. When multiple particles used informed initialization, solutions were drawn from the saved set in order of similarity.

The performance of the algorithm as a function of the number of iterations for the cases when zero, one, and five particles use informed initialization is shown in Figure 5. Using more particles initialized with informed starting positions resulted in faster convergence to lower KL-divergence values. We omitted two, three, and four initialized particles from the graph for clarity, but we noted that there was an incremental decrease in the KL-divergence for each case. We also used a Wilcoxon signed-rank test with a confidence level of 0.95 to compare the results of zero initialized particles to five initialized particles after every 200 iterations. Results indicate that the informed initialization performs significantly better than random initialization at 0, 200, 400, 600, and 800 iterations. Although informed initialization appears to perform better after 1000 iterations as well, the decrease in KL-divergence from standard PSO is no longer statistically significant.

VI. Discussion and Conclusions

The experiment from Section V-A showed that of the optimization methods considered, PSO performed best. In addition, Figure 4 gives a sense of how well phase-type distributions are able to approximate various parameterizations of Weibull and lognormal distributions. KL-divergence values were higher when the true distribution had harsher peaks in its PDF. This indicates that phase-type distributions are better at

approximating smooth distributions. Intuitively, using a larger number of phases should mitigate this problem.

The conclusion to be drawn from the experiment in Section V-B is that a phase-type distribution with a single phase or few phases may lead to unsatisfactory approximations. Six phases seems optimal, as further increasing the number of phases does indeed increase expressiveness, but also makes optimization more difficult. Since the search space increases quadratically with the number of phases, adding more phases greatly expands the parameter space the optimization method must search. We found that three phases is likely sufficient to get a reasonable approximation, and two phases may also work when model complexity is a concern.

As discussed in Section V-C, informed initialization does appear to improve the approximations. Initializing the entire swarm in this way produced significantly better results at intermediate stages of the optimization process. In addition, Figure 5 shows that the solution converges much faster when informed initialization is used, which could be beneficial to applications where learning time is important.

The goal of this work was to develop a method for approximating known parametric distributions using phase-type distributions. We demonstrated that this can be accomplished by using PSO to minimize a modified KL-divergence value. We compared PSO to two baseline algorithms and found that it produced the best approximations. We explored how well this procedure performs for a variety of parameterizations of both Weibull and lognormal distributions, and also tested how the number of phases impacts the quality of the approximation. Finally, we proposed and tested an extension that uses informed initialization to improve convergence speed of the optimization algorithm. Experiments in these areas have shown promising results, paving the way for additional application of PSO and related algorithms to CTBNs.

VII. Future Work

One possible area for future work involves alternate approximations of the matrix exponential. The scaling and squaring method, while accurate, is also computationally expensive. A potential improvement to our approach could be to use a faster approximation initially, and later switch back to the more accurate scaling and squaring. The threshold at which this switch is most effective, along with the specific alternate method used, is left to future work.

We would also like to experiment with approximating distributions beyond Weibull and lognormal. The same framework can be used for any positive distribution, so long as the PDF can be calculated for the KL-divergence calculation. While we expect our results will generalize, it is possible that features unique to other distributions may influence the ability to find satisfactory phase-type approximations.

We plan to provide a more formal treatment of the process required to embed phase-type distributions into a conditional intensity matrix. Although this has been discussed briefly by Nodelman and Horvitz, as well as in later works, there has been no mathematically rigorous explanation given that describes the embedding process. Furthermore, discussion of the embedding process typically assumes certain features of the phase-type distribution, such as an initial distribution that deterministically starts in the first phase.

Finally, we are interested in the effects of moving to a distributed PSO implementation. Several of the papers referenced in the related literature take this approach, and it is likely that a distributed approach would increase the efficiency of our proposed method of parameterizing phase-type approximations of true distributions.

References

[1] U. Nodelman, C. Shelton, and D. Koller, "Continuous time Bayesian networks," in *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002, pp. 378–387.

[2] K. Gopalratnam, H. Kautz, and D. S. Weld, "Extending continuous time Bayesian networks," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20. AAAI Press, 2005, p. 981.

[3] J. Banks, *Handbook of Simulation*. John Wiley & Sons, 1998.

[4] V. T. Farewell and R. L. Prentice, "A study of distributional shape in life testing," *Technometrics*, vol. 19, no. 1, pp. 69–75, 1977.

[5] J. D. Kalbfleisch and R. L. Prentice, *The Statistical Analysis of Failure Time Data*. John Wiley & Sons, 2011, ch. Failure Time Models.

[6] J. T. de Oliveira, "Statistical choice of univariate extreme models," *Statistical Distribution in Scientific Work: Applications in Physical, Social and Life Sciences*, vol. 6, pp. 367–387, 1981.

[7] S. Rajarshi and M. Rajarshi, "Bathtub distributions: a review," *Communications in Statistics - Theory and Methods*, vol. 17, 1988.

[8] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement." in *Proceedings of the 32nd Annual International Symposium on Computer Architecture*. IEEE Computer Society, 2005, pp. 520–531.

[9] O. O. Aalen, "Phase–type distributions in survival analysis," *Scandinavian Journal of Statistics*, vol. 22, no. 4, pp. pp. 447–463, 1995.

[10] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM Review*, vol. 45, no. 1, pp. 3–49, 2003.

[11] N. J. Higham, "The scaling and squaring method for the matrix exponential revisited," *SIAM Journal on Matrix Analysis and Applications*, vol. 26, no. 4, pp. 1179–1193, 2005.

[12] G. A. Baker Jr. and J. L. Gammel, "The Padé approximant and some related generalizations," in *The Padé Approximant in Theoretical Physics*, ser. Mathematics in Science and Engineering. Academic Press, 1970, vol. 71.

[13] U. Nodelman and E. Horvitz, "Continuous time Bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation," *Microsoft Research*, vol. July-August, 2003.

[14] M.-S. Kim, I.-H. Yang, and H.-J. Yu, "Maximizing distance between GMMs for speaker verification," in *Fourth International Conference on Natural Computation*. IEEE, 2008, pp. 175–178.

[15] F. Sahin, M. Ç. Yavuz, Z. Arnavut, and Ö. Uluyol, "Fault diagnosis for airplane engines using Bayesian networks and distributed particle swarm optimization," *Parallel Computing*, vol. 33, no. 2, pp. 124–143, 2007.

[16] N. Fortier, J. Sheppard, and S. Strasser, "Parameter estimation in Bayesian networks using overlapping swarm intelligence," in *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*. ACM, 2015, pp. 9–16.

[17] U. Nodelman, C. R. Shelton, and D. Koller, "Expectation maximization and complex duration distributions for continuous time Bayesian networks," *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005.

[18] J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, no. 2. Perth, Australia, 1995, pp. 1942–1948.

[19] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[20] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vettering, *Combinatorial Minimization: Method of Simulated Annealing*. Cambridge University Press, 1986.