Preventive Maintenance

Martijn Gösgens

September 18, 2017

1 Abstract

Contents

2 Introduction

3 LiteratureOverview

4 Simplified models

As a first attempt to investigate the relevant literature, as well as the underlying complexity of the underlying methodological aproach, we begin with a simple mathematical model before we incrementally add complexities to make it more realistic. Besides the above mentioned advantages, we envision that such a simplification will permit us to... TODO

4.1 First model

In our first and simplest model, we define the problem as a renewal process where the machine is renewed after it is replaced. The machine has a lifetime L that follows some continuously differentiable distribution $f_L(l)$ (cumulative function $F_L(l)$). Replacing the machine has a certain cost $c > 0$. When the machine breaks, it needs to be repaired and a certain additional cost for the downtime $A > 0$ needs to be paid such that repairing the machine correctively has a cost $c + A$. Repairing the machine takes zero time. We start with a continuous time problem, i.e. we may choose any time at which the machine will be preventively repaired. When the machine is repaired, the problem starts again with a new lifetime according to the same distribution.
We want to minimize the average cost. Since the machine is renewed after each repair, we can do this by minimizing the expected average cost until the first repair.

Imposing a policy of control-limit type, we will look for a policy where we decide to preventively repair the machine at some time $u > 0$, then with probability $1 - F_L(u)$ a cost of c will occur after time u and for each time $l < u$ the probability density a cost $c + A$ occurring is $f_L(l)$. This results in the expected average cost of

$$J(u) = \frac{(1 - F_L(u))c}{u} + \int_0^u \frac{f_L(l)(c + A)}{l} dl.$$

We want to minimize this. To find this minimum, we first try the extreme values. For $u = 0$, there would be an infinite cost per time unit so this is clearly not minimal. $u = \infty$ is equivalent to not doing any preventive maintenance ($F_L(\infty) = 1$ so the first term vanishes). As you can see, the expected cost then equals $(c + A)\mathbb{E}[1/L]$. The minimum can also be at a zero of the derivative of the cost function:

$$\frac{d}{du} J(x) = -\frac{c}{u^2} - \frac{f_L(u)c}{u} + \frac{F_L(u)c}{u^2} + \frac{f_L(u)(c + A)}{u} = 0.$$

We multiply by u^2 ($u > 0$ as preventive maintenance at time 0 would result in infinite average cost):

$$-c(1-F(u))+uf_L(u)A=0. \tag{1}$$

Transforming this, we can recognise the hazard rate function in it

$$H(u):=\frac{f(u)}{1-F(u)}=\frac{c}{uA}. \tag{2}$$

Where $J(u)$ is decreasing iff $H(u)<\frac{c}{uA}$ and this inequality holds for sufficiently small u.Hence, if the hazard rate decreases less than $\frac{c}{uA}$ for sufficiently large u, the optimal stopping time would be one of the solutions of (2).

Check edge cases where $L=0$

Example 4.1. For example, for lifetimes uniformly distributed on the interval $[0,B]$, this would result in solving

$$\frac{1}{B-u}=\frac{c}{uA}\Rightarrow uA=c(B-u)\Rightarrow u=\frac{cB}{c+A}.$$

Example 4.2. For a Weibull distributed lifetime with scale λ and shape k, filling in (2) results in

$$\frac{ku^{k-1}}{\lambda^k}=\frac{c}{uA}\Rightarrow u=\lambda\sqrt[k]{\frac{c}{kA}}.$$

For an exponentially distributed lifetime ($k=1$) with rate $\lambda^*=1/\lambda$ this results in

$$u=\frac{c}{\lambda^* A}.$$

4.2 Discounted model

The previous calculation could also have been done using a continuous discount α such that costs occurring at some time $t>0$ are discounted by a factor $e^{-\alpha t}$. The expected cost of repairing at time $u>0$ is

$$J(u)=e^{-\alpha u}(c+J(u))(1-F_L(u))+(c+A+J(u))\int_0^u f_L(t)e^{-\alpha t}dt. \tag{3}$$

We can rewrite this to

$$(1-e^{-\alpha u}(1-F_L(u))-\int_0^u f_L(t)e^{-\alpha t}dt)J(u)=c(e^{-\alpha u}(1-F_L(u))+\int_0^u f_L(t)e^{-\alpha t}dt)+A\int_0^u f_L(t)e^{-\alpha t}dt.$$

When we additionally notice that

$$e^{-\alpha u}(1-F_L(u))-\int_0^u f_L(t)e^{-\alpha t}dt=\mathbb{E}(e^{-\alpha\min\{L,u\}})$$

and that

$$\int_0^u f_L(t)e^{-\alpha t}dt=\mathbb{E}(\mathbb{1}\{L<u\}e^{-\alpha L}).$$

Then, we can rewrite the discounted cost function to

$$\begin{aligned} J(u) &= \frac{c\mathbb{E}(e^{-\alpha \min\{L,u\}}) + A\mathbb{E}(\mathbb{1}\{L<u\}e^{-\alpha L})}{1-\mathbb{E}(e^{-\alpha \min\{L,u\}})} \\ &= \frac{c + A\mathbb{E}(\mathbb{1}\{L<u\}e^{-\alpha L})}{1-\mathbb{E}(e^{-\alpha \min\{L,u\}})} - c. \end{aligned} \tag{4}$$

We want to minimize this. Note that this minimum is again not at $u = 0$ since then

$$J(u) = c + J(u).$$

The minimum could be at $u = \infty$. The resulting cost would be

$$J(u) = (c + A + J(u))\mathbb{E}(e^{-\alpha L}) \Rightarrow J(u) = \frac{(c+A)\mathbb{E}(e^{-\alpha L})}{1-\mathbb{E}(e^{-\alpha L}}.$$

The minimum could also be a zero of the derivative of (3):

$$\begin{aligned} 0 = \frac{d}{du}J(u) &= -\alpha e^{-\alpha u}(c+J(u))(1-F_L(u)) - f_L(u)e^{-\alpha u}(c+J(u)) \\ &\quad + e^{-\alpha u}\frac{d}{du}J(u)(1-F_L(u)) \\ &\quad + f(u)e^{-\alpha u}\frac{d}{du}J(u) + (c+A+J(u))f_L(u)e^{-\alpha u} \\ &= -\alpha e^{-\alpha u}(c+J(u))(1-F_L(u)) + Af_L(u)e^{-\alpha u} \\ &\Rightarrow H(u) = \frac{\alpha(J(u)+c)}{A}. \end{aligned} \tag{5}$$

Or, using (4), a solution of

$$H(u) = \frac{\alpha}{A}\frac{c + A\mathbb{E}(\mathbb{1}\{L<u\}e^{-\alpha L})}{1-\mathbb{E}(e^{-\alpha \min\{L,u\}})}.$$

4.3 Discretized time

As in the real world the machine cannot be repaired at each instance, it would be interesting to discretize the time as $t_k = k\Delta$. There is one state (s_0). At this state, there are two actions:

Replace cost notation.

- Preventive maintenance (u_p) with cost c_p, renewing the machine.
- Do nothing (u_w) with cost 0 if the machine does not break and cost c_c (for corrective maintenance) if the machine does break. Corrective maintenance renews the machine.

Let B_k denote the event that the machine breaks between stages k and $k+1$ and let $p_k = \mathbb{P}(B_k)$. For simplicity we introduce a discount α and are interested in the discounted cost instead of the average cost. When we choose to preventively repair the machine, a cost c_p will be paid and the machine will be renewed in the next step. If we do not repair the machine, no immediate cost is paid and we will have a chance of p_k of having to pay the cost for corrective maintenance and renewing the machine in the next step and a probability of $1-p_k$ of going to the next step without the machine failing. The value function would then be

$$J_k(s_0) = \min\{c_p + \alpha J_0(s_0), (1-p_k)\alpha J_{k+1}(s_0) + p_k\alpha(c_c + J_0(s_0))\}.$$

4.4 Continuous Time Markov Chain Model

If, in the previous problem, p_k would be the same for each k, this would be equivalent to a Discrete Time Markov Chain with one state and a failure probability p_k at each step. In this section we will extend this to general CTMCs where each state has some failing rate. This is useful since the lifetime of the machine would then be of Phase Type distribution and these are dense in the class of positive continuous distributions[5]. We assume that the machine behaves as a continuous time Markov chain with $n+1$ states with an initial distribution $p_0(i)$ for $i = 1, ..., n$, where a failure occurs when a transition to some fail state $s_F = s_{n+1}$ is made. We assume perfect state information and the action of repairing the machine (u_r) can be taken at any time. The rest of the time, the action u_w is chosen. Preventive maintenance and corrective maintenance have again costs c_p and c_c. These costs occur immediately when deciding preventive maintenance or repairing the broken machine. After maintenance, the machine is renewed and enters in one of the initial states according to the initial distribution. We introduce continuous discount of the form $e^{-\beta t}$.

Making decisions when entering the state only

Since having spent some time in one state does not change anything about the transition probabilities, an optimal policy cannot choose preventive maintenance in between transitions if it did not choose to do so when entering the state. Hence, we only consider policies where decisions are made when entering a state.

Uniformization

We can then transform this problem to a decision problem with a discrete time Markov chain and a discount factor $\alpha = \frac{\nu}{\nu+\beta}$ (where ν is an upper bound for the transition rates) by uniformization[17]. The adjusted transition probabilities would then be[17]:

$$\tilde{p}_{ij}(u_w) = \frac{\nu_i u_w}{\nu} p_{ij}(u_w) + \delta_{ij}(1 - \frac{\nu_i}{\nu}),$$

where $\delta_{ij} = 1$ if $i = j$ and zero else. The Bellman equations would then be for $i = 1, ..., n$:

$$J(i) = \min\{c_p + \sum_{j=1}^{n} p_0(j)J(j), \alpha \sum_{j=1}^{n+1} p_{ij}(u_w)J(j)\}$$

and for $i = n+1$:

$$J(n+1) = c_c + \sum_{j=1}^{n} p_0(j)J(j).$$

Note that the costs in the failed state and for the preventive maintenance action are not multiplied by the discount factor, in order to make the repairs happen instantaneously.

Implementation

We used the event log of the machine to extract a continuous time Markov chain. We then implemented the approach described above together with the Gauss-Seidel version of value iteration[3]. This was chosen because it is easy to implement and because the extra computation time it takes compared to policy iteration is no problem in the current setting. The choice for the Gauss-Seidel

process feedback

version was made because it converges slightly faster than the original Value Iteration algorithm and because it is easier to implement. The value iteration is stopped after 10000 iterations, which resulted in error bounds lower than 10^-5 in the observed cases.

Add results

4.5 Fluid models

The previous problem could be seen as a Markov modulated fluid model with one state with rate -1 and a random initial fluid level according to distribution $F_L(l)$. We could extend this to fluid models of more states. We introduce a Markov modulated fluid model with two states:

- s_0: With fluid rate $r_0 < 0$
- s_1: With fluid level $r_1 < 0$.

The system transitions in the fluid model occur as a CTMC when u_w is chosen and the machine does not break, when u_p is chosen or the machine breaks (and corrective maintenance is performed), the system is renewed and transitions to s_0. The initial fluid level is again given by the distribution $F_L(l)$.

Since, for the Bellman Equations, we need to have the transition probability densities for the fluid levels. To calculate the probability that the machine breaks between two stages, we need to have the distribution of the fluid decrease in a period of length Δ. Let $\overline{r} = \max\{r_0, r_1\}$ and $\underline{r} = \min\{r_0, r_1\}$. Let $\Delta\underline{r} < q < \Delta\overline{r}$, then the probability that the fluid decreases more than q in the next period, equals the probability that the machine spends less than $\frac{\overline{r}\Delta - q}{\overline{r} - \underline{r}}$ time in the state with the lowest rate. Let $f_i^j(t^*, t)$ denote the density of spending t^* out of t time in state j given that it starts in state i. We have that for small h:

$$f_0^1(t^*, t) = \lambda_0 h f_1^1(t^*, t-h) + (1 - \lambda_0 h) f_0^1(t^*, t-h)$$

$$\Rightarrow \frac{f_0^1(t^*, t) - f_0^1(t^*, t-h)}{h} = \lambda_0 (f_1^1(t^*, t-h) - f_0^1(t^*, t-h))$$

And as $h \to 0$ this results in

$$\frac{d}{dt} f_0^1(t^*, t) = \lambda_0 (f_1^1(t^*, t) - f_0^1(t^*, t))$$

Similarly, for $f_1^1(t^*, t)$ we have

$$(\frac{d}{dt} + \frac{d}{dt^*}) f_1^1(t^*, t) = \lambda_1 (f_0^1(t^*, t) - f_1^1(t^*, t))$$

It is difficult to solve these equations, therefore we apply a probabilistic approach and check whether the results adhere to the differential equations afterwards. Let $T_0(t)$ denote the random variable of the amount of time spent in state 0 out of total time t. Likewise $T_1(t) = t - T_0(t)$. Furthermore, Let $N(t)$ denote the total number of transitions that occurred between time zero and t. (Note: in the above definitions, we assumed that the model is in state 0 at time 0) If $N(t)$ is even, then the same number of transitions from s_0 to s_1 occurred as from s_1

to s_0. If $N(t)$ is odd, one transition must have occurred from s_0 to s_1 than from s_1 to s_0. Now we have

$$\begin{aligned} f_0^0(t^*,t) &= \lim_{h\to 0}\frac{1}{h}\mathbb{P}(t^* \le T_0(t) < t^*+h) \\ &= \lim_{h\to 0}\frac{1}{h}\mathbb{P}(t^* \le T_0(t) < t^*+h, \bigcup_{n=0}^{\infty} N(t)=n) \\ &= \lim_{h\to 0}\frac{1}{h}\sum_{n=0}^{\infty}\mathbb{P}(t^* \le T_0(t) < t^*+h, N(t)=n) \\ &=: \sum_{n=0}^{\infty} g_0^{(n)}(t^*, t-t^*) \end{aligned} \tag{6}$$

Where $g_0^{(n)}(t_0,t_1)$ denotes the density of n transitions occurring and spending t_0 out of t_0+t_1 time in s_0. If $t_1>0$, then $g_0^{(n)}(t_0,t_1)=0$ for $n>0$. If n is even, then the system ends in s_0, else it ends in s_1. If the system ends in s_0, this means that the $n/2$-th transition from s_1 to s_0 occurred exactly after it has spent t_1 (total) time in s_1, while exactly $n/2$ transitions occurred from s_0 to s_1 in the t_0 time it has spent in s_0. This corresponds to the probability of $n/2$ arrivals of a Poisson process with rate $\lambda_0 t_0$ multiplied with the density of an Erlang distributed variable with rate λ_1 and shape $n/2$ at time t_1.
If the system ends in s_1, this means that exactly $(n-1)/2$ transitions from s_1 to s_0 occurred in the time t_1 it has spent in s_1 while the $(n+1)/2$-th transition from s_0 to s_1 occurred after spending t_0 time in s_0. This corresponds to the distribution of an Erlang distributed variable with rate λ_0 and shape $(n+1)/2$ at time t_0 multiplied by the probability of $(n-1)/2$ arrivals of a Poisson process with rate $\lambda_1 t_1$.
We introduce some notation: Let P_λ be a Poisson distributed variable with rate λ, let $E_{\lambda,n}$ be an Erlang distributed random variable with rate λ and shape n and let $f_x(t)$ be the density of some random variable X. Then we can write:

$$\begin{aligned} g_0^{(2k)}(t_0,t_1) &= \mathbb{P}(P_{\lambda_0 t_0}=k) f_{E_{\lambda_1,k}}(t_1) \\ &= \frac{(\lambda_0 t_0)^k e^{-\lambda_0 t_0}}{k!}\frac{\lambda_1^k t_1^{k-1} e^{-\lambda_1 t_1}}{(k-1)!} \\ &= e^{-\lambda_0 t_0-\lambda_1 t_1}(\frac{(\lambda_0\lambda_1 t_0)^k t_1^{k-1}}{k!(k-1)!} \end{aligned}$$

and

$$\begin{aligned} g_0^{(2k-1)}(t_0,t_1) &= f_{E_{\lambda_0,k}}(t_0)\mathbf{P}(P_{\lambda_1 t_1}=k-1) \\ &= \frac{\lambda_0^k t_0^{k-1} e^{-\lambda_0 t_0}}{(k-1)!}\frac{(\lambda_1 t_1)^{k-1} e^{-\lambda_1 t_1}}{(k-1)!} \\ &= \frac{\lambda_0^k(\lambda_1 t_0 t_1)^{k-1}}{(k-1)!^2}) \end{aligned}$$

The density $f_0^0(t^*,t)$ can then be obtained by summing these (assuming $t^*<t$ and hence $g_0^{(0)}(t^*,t-t^*)=0$:

$$f_0^0(t^*,t) = \sum_{n=1}^{\infty} g_0^{(n)}(t^*,t-t^*)$$

And we have that $f_0^1(t^*, t) = f_0^0(t - t^*, t)$. Moreover, we can obtain $f_1^1(t_0, t_0 + t_1)$ by interchanging λ_0 with λ_1 and interchanging t_0 with t_1 in the expression of $f_0^0(t_0, t_0 + t_1)$.

It can be seen that these expressions adhere to the differential equations mentioned earlier by simply filling them in.

Theorem and proof environments and connect to a solution

5 Preliminary Data Analysis

In this section, the given data will be explored and visualised.

5.1 Hazard Rate

Before trying to find a policy to optimally schedule preventive maintenance, it would be useful to first consider whether maintenance is helpful at all. To do this, we can consider the hazard rate of the machine. An increasing hazard rate results in a decreased expected lifetime over time while a decreasing hazard rate results in an increased expected lifetime. Below, you see a plot of this.

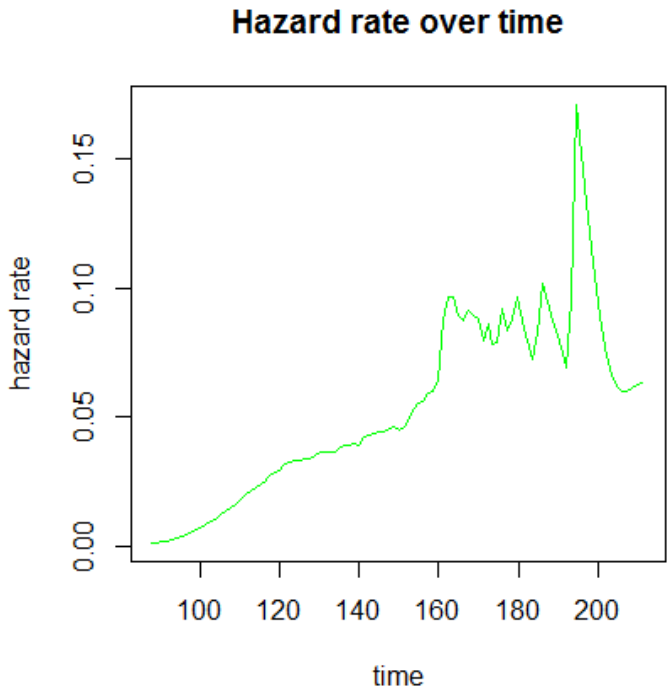

Figure 1: Hazard rate of the machine under consideration. The hazard rate was estimated by first estimating the probability density *pdf* and cumulative distribution function *cdf* using standard R functions and then calculating $\frac{pdf}{1-cdf}$.

The hazard rate seems to be decrease after 200 hours, but this can be explained by the fact that there is only one trace that lasted over 200 hours.

Hence, we can conclude that the hazard rate rises over time (at least for the first 200 hours). Planning preventive maintenance is only useful for increasing hazard rates. If the hazard rate would decrease, the expected lifetime of the machine would increase over time, making renewing the machine counterproductive. If the hazard rate would be constant then the expected lifetime would also be constant and renewing the machine would have no effect.

5.2 Time Until Failure

To be able to predict the remaining time until a failure, it is helpful to know how the total lifetime of the machine is distributed. Of course, there is a one to one correspondence between the hazard rate and the distribution of the lifetime. In this section we will attempt to fit the lifetime to a distribution.

5.2.1 Cullen and Frey graph

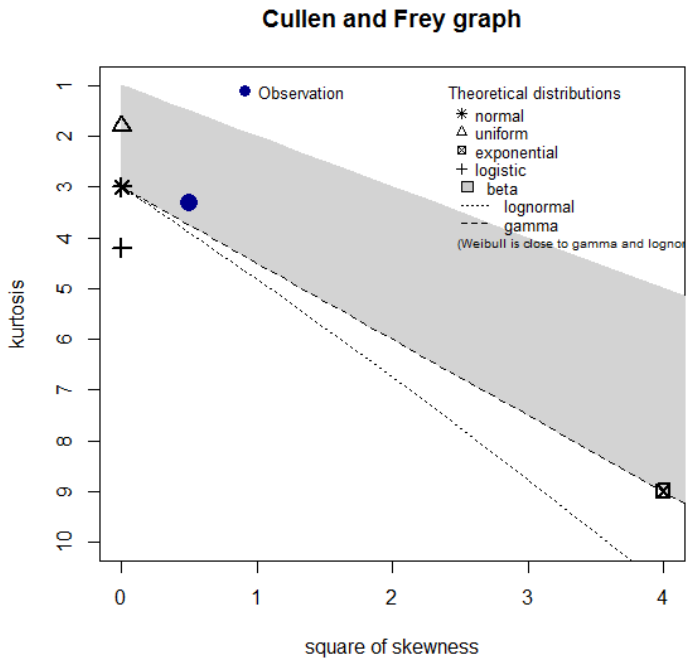

Figure 2: Cullen and Fray graph of the data.

Above, a Cullen and Fray graph is plotted. The data is placed based on its kurtosis and skewness. A few well-known distributions are also plotted on this plane. This plot would suggest a beta distribution. However, the beta distribution has a support of $[0, 1]$ while the lifetime is not bounded (Although the machine has only been in place a limited time). Other distributions that have similar kurtosis and skewness are the Weibull distribution and the gamma distribution.

Write Appendix about Cullen and Fray graphs.

After estimating the parameters for each of these distributions and performing a Anderson-Darling Goodness-of-Fit test, the gamma-distribution seems to fit the data best with a p-value of 0.195 versus a p-value of 0.00444. Below, a plot shows the density of this distribution plotted over the observed density. As you can see, it does not fit the data very well.

Explain and justify in Appendix.

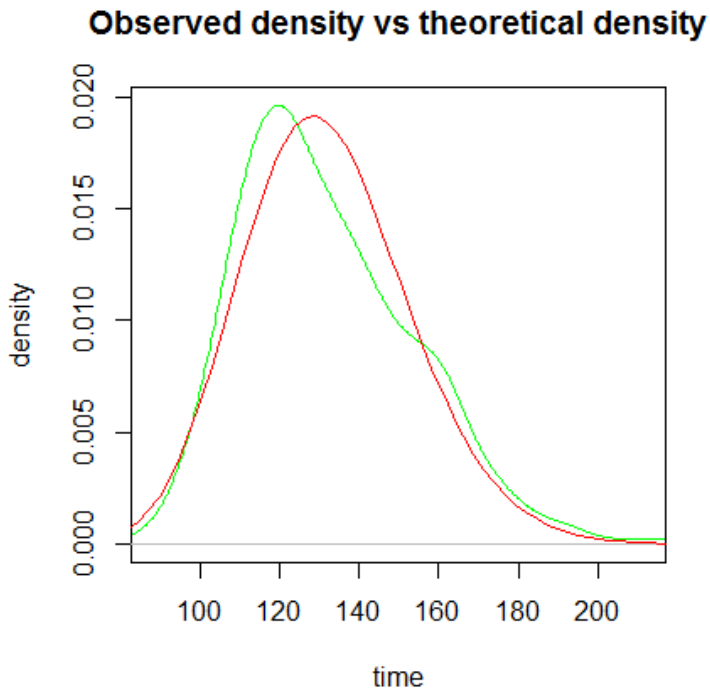

Figure 3: The observed density plotted over the density of the Gamma distribution

5.2.2 Phase-Type distribution

As the class of Phase-Type distributions is dense in the space of positive continuous distributions, a Phase-Type distribution could also be used to model the lifetimes. However, Phase-Type distributions have a few disadvantages: The number of parameters grows quadratically with the amount of states and a lot of these parameters are redundant. Furthermore, convergence of the EM-algorithm (to estimate the parameters) is slow and can get stuck in saddle points and local maxima [1].

5.2.3 Transition Times

If we view each event as a state and view switching from one event to another as a transition. Then we could model the machine as a transitioning system. It would then be interesting to see how the transition times are distributed; whether they are of the same family of distributions and whether they are

independent.
When using the Anderson-Darling Goodness-of-Fit test for testing whether the transition times are exponentially distributed, the resulting p-values are rather low.

Quantify

6 Fluid models

Add introduction defining markov modulated fluid models and outlining relevant theory

6.1 Rate estimation

Given a CTMC $X(t)$ with states s_i for $i = 1, ..., n$ and infinitesimal generator matrix Q, we want to predict some event using a Fluid Model with fluid level $L(t)$ with initial distribution $F_L(q)$ and density $f_L(q)$, we want to find the rates $r_i \leq 0$ for each state s_i.
Let X_k denote the k-th state that the CTMC visits, let T_k denote the time the CTMC spends in this state and let F_k denote the occurrence of the event while the CTMC is in its k-th state. We assume that the event occurs when the fluid level reaches zero.
We then have

$$\mathbb{P}(F_k|X_k = s_i) = \int_0^\infty \mathbb{P}(T_k > \frac{l}{-r_i}) f_{L_i}(l)dl = \int_0^\infty e^{-q_{ii}l/r_i} f_{L_i}(l)dl$$

Where f_{L_i} denotes the density of the fluid level when the CTMC arrives in s_i. Given some density function f_{L_i}, we can use this to make a maximum likelihood estimator for r_i. Also another relation holds between the fluid levels of different states[9]:

$$\frac{\partial}{\partial t} p_i(t, l) + \frac{\partial}{\partial l} r_i p_i(t, l) = \sum_{k=1}^{n} q_{ki} p_k(t, l)$$

6.2 Finding jump states

Some transitions that the machine makes, are beneficial for its lifetime. To model this behavior, we can add jump transitions. Certain transitions could add a certain (stochastic) amount to the fluid quantity. We define a jump state as a state for which each leading transition is a jump transition. For simplicity, we will start by looking for jump states.

Add examples

As a jump transtion increases the fluid quantity and a higher fluid quantity is beneficial for the machine, a jump transition will likely increase the expected lifetime. Hence, we call a transition a jump transition when its source state has a lower expected time to live than the destination state. A jump state is then a state which has a greater time to live than each of its incoming states.

We implemented this method by calculating the expected lifetime in each state (in each state we took the average of the time until the end of the trace

for each occurrence). And then we compared this to the incoming states. This resulted in finding state 1 and 39 as jump states.

Of course, this method should mostly be seen as a heuristic for finding interesting states rather than a way to prove that these states are jump states.

7 Technical report

During this project, several tools were used. In this section, it will be described how these tools were used. The encountered technical difficulties and their solutions or workarounds will also be reported.

7.1 ProM

ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins[**?**]. During this project, ProM is used to analyse the event logs. Since it might be that not every process mining technique that is useful for this project is already implemented in ProM, I figured it would be useful to be able to create my own.

7.1.1 Plug-in development

To learn how to make such plug-ins, I found the following tutorial by M. Westergaard very helpful:
`https://westergaard.eu/category/prom-tutorial/`
Naturally, this short tutorial did not cover all information necessary to be able to deploy ProM plug-ins. Hence, I also had to occasionally consult ProM's Javadoc and forum. I used the Eclipse IDE to write the Java code for the plugin. There was some difficulty in being able to load the plugins into ProM since the ProM package manager gets the available plugins from a number of repositories on the `promtools.org` website and I was not able to add my own plugins to those repositories. After some research, trial and error, I came to the following workaround: The ProM package manager keeps a list of repositories in an xml-file that is locally stored in the 'packages'-folder. This file ('packages.xml') directs to a number of remote xml-files (also named 'packages.xml') which refer to zip-files which contain the jar-file of the plugin and a folder 'lib' which contains the jar-files of the libraries that the plug-in needs to be able to run. After downloading one of those remote xml-files, I figured out the format of these files and created a local packages.xml-file for my own plugins to which I referred to in the first packages.xml file as a new repository. I also exported my project as a jar-file and zipped it together with the required libraries. This allowed me to install my own package using the ProM package manager. The downside of this method is that it is very time-consuming to do all of this each time the plug-in is changed. Luckily, Eclipse also allows for running the plug-ins in debug mode. The downside of using debug mode, however, is that other plug-ins are not available unless you have imported their containing packages as libraries in the Eclipse project, and for certain plug-ins (mostly export and import plug-ins) it is difficult to find out to which package they belong. To workaround this issue, I created my own xes-importer so that I could test the

plug-ins in debug mode. But since the full event-log of this project is rather large (over 3GB) and my basic xes-importer was not disk-buffered, I was only able to test the plug-ins on a small subset of the data in debug mode. Another problem that I encountered, is that some basic functions (for instance putIfAbsent() from java.util.HashMap) only worked in debug mode. This was hard to find out, but could easily be worked around by simply not using this function.

7.1.2 Plug-ins created

Below is a list of the various plug-ins that were created during this project together with a short description:

- **XesImportPlugin**: This plug-in imports xes-files, it was made to work around the fact that the existing importers were not available in debug mode.
- **DiscreteTimeMarkovChainPlugin**: This plug-in takes a XLog and produces a DiscreteTimeMarkovChain, using the events from the log as states.
- **StationaryDistributionPlugin**: This plug-in takes a DiscreteTimeMarkovChain and computes the eigenvalue-decomposition of the transition-matrix, from which the stationary distribution can be read. The output is a StationaryDistribution object.
- **StationaryDistributionExportPlugin**: This plug-in exports a StationaryDistribution object to a txt-file.
- **TraceLifetimesPlugin**: This plug-in takes an XLog as input and returns a list with the lifetimes of the traces. This list is encapsulated by an TraceLifetimes object.
- **TraceLifetimesExportPlugin**: This takes a TraceLifetimes object as input and stores the lifetimes in a csv-file so that they can easily be analysed using an R-script.

7.2 Event log

The data set that was analysed was a XES-file. XES stands for eXtensive Event Stream and is a format for event log files. There were some issues when reading the data: The event logs had timestamps for the start and end-time of each transition. When inspecting the data more closely, I found out that there are some transitions occurring in negative time. Most of them had a length of -1 seconds, which is likely caused by rounding errors, but there were also two outliers; two transitions with length almost minus 3600 seconds. These both occurred on the last sunday of March around 3 o'clock in the night but in different years. This is of course the time that the clock is moved an hour forward to transition from winter to summer time. The most likely explanation is that the event logger of the machine did not use summer and winter time while the OpenXES framework that was used to extract the timestamp from the date-times did. This error was discovered for the transition to summer time but the clock is also moved back each year, which will likely cause an error as well.

7.3 R

R is a scripting language for statistics. During this project, it has been used to perform statistical analysis on data extracted from the event logs.

8 Literature

- [1] Discusses the Phase-Type distribution and methods to estimate the parameters.
- [3] Book on dynamic programming and optimal control.
- [2] Introduces a new policy iteration-like algorithm for finding the optimal state costs or Q-factors and proves convergence properties.
- [4]
- [5] Concerns the Phase-Type distribution. Shows properties and proposes methods to estimate the parameters. Mentions the difficulties: Many parameters (n^2 for n states), of which, most are redundant but in most cases there is no canonical representation for each class ($2n-1$ independent parameters).
- [6] Uses BIDE to mine closed frequent sequential patterns.
- [8] A Survey of Sequential Pattern Mining. Describes general concepts, problem variants and the existing algorithms.
- [9] Intoduces Reward Model, Fluid Model (first and second order, reflecting and absorbing boundary) and Fluid Stochastic Petri Nets. Gives differential equations for transient behavior and proposes methods to solve/compute them.
- [10] Estimates transition probabilities for countably infinite state spaces.
- [11] Concerns Fluid Stochastic Petri Nets, summarizes the theory and provides examples.
- [12] Concerns estimating CTMC's with discretely observed data.
- [13] Considers three states (perfect, satisfactory and failed) and maintenance at scheduled (fixed interval) and unscheduled (exponential interval) times and finds the policy that minimizes the cost.
- [14] Proposes a method of modelling insurance losses via mixtures of Erlang distributions. Also explains why Phase-type distributions are not practical for this situation.
- [16] Models software faults with intervals of phase type distribution.
- [19] Introduces jumps that can occur at transition times, the size of the jumps are according to some distribution. The paper gives the resulting differential equations and methods to solve them.

- [20] Introduces the BIDE algorithm for finding closed frequent sequential patterns.
- [18] Proves that for every CTMC decision problem (where decisions are made at the beginning of the sojourn time) there is an equivalent DTMC decision problem.
- [15] (Section "Markov Chains and Spectral Clustering") Regards spectral clustering, shows that paritioning the state to minimize the cut is equivalent to minimizing to maximize the internal cohesion of the clusters. Clusters can be formed by grouping on the entries of the eigenvectors corresponding to the subdominant eigenvalues. Clusters can be made by grouping on multiple eigenvectors or by iteratively partitioning a cluster on the subdominant eigenvector.
- [7] Introduces the MCL algorithm for clustering graphs. Focusses mainly on undirected graphs and there are not many characteristics of the resulting clusters proven.

References

[1] Søren Asmussen, Olle Nerman, and Marita Olsson. Fitting phase-type distributions via the em algorithm. *Scandinavian Journal of Statistics*, pages 419–441, 1996.

[2] Dimitri P Bertsekas and Huizhen Yu. Q-learning and enhanced policy iteration in discounted dynamic programming. *Mathematics of Operations Research*, 37(1):66–94, 2012.

[3] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[4] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[5] Peter Buchholz, Jan Kriege, and Iryna Felko. *Input modeling with phase-type distributions and Markov models: theory and applications*. Springer, 2014.

[6] Jun Chen and Ratnesh Kumar. Pattern mining for predicting critical events from sequential event data log. *IFAC Proceedings Volumes*, 47(2):1–6, 2014.

[7] S VAN Dongen. *Graph clustering by flow simulation*. PhD thesis, Standardization and Knowledge Transfer, 2000.

[8] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and R Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[9] Marco Gribaudo and Miklós Telek. Fluid models in performance analysis. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 271–317. Springer, 2007.

[10] Monir Hajiaghayi, Bonnie Kirkpatrick, Liangliang Wang, and Alexandre Bouchard-Côté. Efficient continuous-time markov chain estimation. In *ICML*, pages 638–646, 2014.

[11] Graham Horton, Vidyadhar G Kulkarni, David M Nicol, and Kishor S Trivedi. Fluid stochastic petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105(1):184–201, 1998.

[12] Yasunari Inamura et al. Estimating continuous time transition matrices from discretely observed data. Technical report, Bank of Japan, 2006.

[13] Szilard Kalosi, Stella Kapodistria, and Jacques AC Resing. Condition-based maintenance at both scheduled and unscheduled opportunities. *arXiv preprint arXiv:1607.02299*, 2016.

[14] Simon CK Lee and X Sheldon Lin. Modeling and evaluating insurance losses via mixtures of erlang distributions. *North American Actuarial Journal*, 14 (1):107–130, 2010.

[15] Ning Liu and William J Stewart. Markov chains and spectral clustering. In *Performance Evaluation of Computer and Communication Systems. Milestones and Future Challenges*, pages 87–98. Springer, 2011.

[16] Hiroyuki Okamura and Tadashi Dohi. Building phase-type software reliability models. In *Software Reliability Engineering, 2006. ISSRE'06. 17th International Symposium on*, pages 289–298. IEEE, 2006.

[17] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[18] Richard F Serfozo. Technical notean equivalence between continuous and discrete time markov decision processes. *Operations Research*, 27(3):616–620, 1979.

[19] Elena I Tzenova, Ivo JBF Adan, and Vidyadhar G Kulkarni. Fluid models with jumps. *Stochastic models*, 21(1):37–55, 2005.

[20] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 79–90. IEEE, 2004.