University College Cork

Master's Thesis

Process Mining

Richard Collins

supervised by
Dr. Steve Prestwich
2019-09-01 Sun

Declaration

I, Richard Collins, declare that all work presented within is my own and where the information has been taken other sources, those papers or websites have been cited.

Acknowledegements

I have some people to thank.
I would like to thank Dr. Steve Prestwich for agreeing to take on this project with me, and providing advice, support, and experience for it's duration.
I would like to thank Martin Brown, Dhanya Jayachandra and Madhan Balasubramanian at Unitek.BPI for the ideas and inspiration and data to work with.
I would like to thank the my friends and my family for your herculean support.
And you. For reading this. Thank you.

Contents

1 Abstract

A common problem in process mining is that of finding out exactly how long a process should take by decomposing a process into a series of different activities. This work hypothesizes that the time it takes to complete a process is simply the sum of the times to complete the activities in that process. This hypothesis should hold in memoryless processes but does not hold without memorylessness. Two datasets are used in this work

- a dataset provided by `PM4PY` Github Link
- a dataset provided kindly by Unitek BPI.

The latter is proprietary data used to evaluate the model on a real world computational processes. Models are built as specified and fit on data, where it is shown using hypothesis tests and graphical methods to not be appropriate due to violations of the memorylessness property.

2 Introduction of project and background literature

2.1 Literature review

The majority of the literature used in this project exists on an axis with process mining and business process management work on one side, and statistical analysis of markov chains and stochastic processes on the other. The goal of the project was to investigate process mining as a field and to investigate how to use stochastic processes and statistical models to enhance models found through process mining.

A good primer on process mining can be found in [1]. The foundational advances in process mining that much of this work is based on is the work by W. Van der Aalst et. al. in [1], introducing the α algorithm and the notion of representing workflows as a workflow net as characterised in [2] . There now exist many ways to create Petri nets and process models from process logs created by enterprise resource planning (ERP) systems and other sources of lists of events - known as event logs. Much of the functionality used in this project come from [3] and references to the algorithms used are included. `PM4PY` is an open source python library that allows for process mining to be performed in python. While much functionality is provided in this library, the analysis of stochastic petri nets in the package is limited to the memorylessness SPN (stochastic petri net) construction where a memoryless firing policy is used, allowing the petri net to be represented as a Continuous Time Markov Chain (CTMC) as detailed in [4]. The construction of this net is detailed in [5], using the process tree construction detailed in [6] and [7]. A central challenge in this investigation was how to answer the problem of "is this a reasonable way to assess the time evolution of the process?". While techniques exist for building generalised stochastic petri nets with general distributions, (see the GSPN construction in [5]) the memoryless construction is a much more simple model and can be used to considerable effect in understanding the broad time evolution of a process. This project details the effects of fitting Continuous time Markov Chain to a workflow net discovered by the IM_D algorithm very similar to what is described in [8]. This uses the Continuous Time Markov chain construction from a stochastic workflow net. In using this construction, restrictions are placed on the data that can be tested statistically using hypothesis tests. The results of these tests can be extrapolated from to rule out memorylessness in a process on which a memoryless stochastic petri net has been constructed.

2.2 A note on PM4PY

Process mining and data mining are two similar disciplines separated by tooling. For the most part process mining is performed in proprietary software or the `PROM` framework, an open source framework detailed in [9]. Used in this work is a python library called `PM4PY` which is detailed in [3] and allows process mining to be done in a Python environment, more similar to tools familiar to data scientists such as `pandas` [10], `scikit-learn`, and `scipy` [11].

3 Background theory

3.1 What is process mining?

Process mining is a discipline of data mining that involves the extraction of information from sequential event logs. The event log is a series of categorised and sorted events that would happen over the course of a particular period of time. These events known as activities are categorised into cases and sorted by time. Generally speaking activities repeat a number of times in the dataset but may only occur once or twice over a particular case Event logs are comprised of three pieces of information:

1. the activity ID
2. the case ID
3. the timestamp.

More information may be available but this is what fundamentally comprises an event log the core data structure involved in process mining.

3.1.1 An analogy.

Imagine analysing the activity of a baker in a bakery and being asked to categorise everything the baker does in order to create a cookbook. In this case, a cookbook being a list of every possible way this baker could feasibly create a cake The act of making a cake is undoubtedly a complex and variable process considering the amount of possible ways to make what could be called a cake (even just in the English language!), with multiple different permutations of activities involved in the specific case dependant on

what kind of cake being made. By recording and categorising each of the bakers specific activities (for example adding ingredients to a mixing bowl, mixing ingredients, kneading dough, cooking dough in oven for a specified amount of time etc, placing mixtures in the fridge to set overnight) and recording when they happen and how long it takes, an event log would be formed of these activities. The activity IDs would tend to be the name of the activity meanwhile the case IDs maybe take may take the form of a hash or UUID. Each case ID would specify each act of creating a cake that could be described by a concatenation of multiple activities.

3.1.2 What kinds of process mining is there?

Three activities tend to fall under the umbrella of process mining

Process Discovery this would be the act of using algorithms to construct a model that could constructs and replicate existing processes in the event log. in this work the inductive mining technique is used to create a directly follows graph which can be converted into a workflow net, a particular class of petri net with useful properties which can be converted into a transition diagram from which the time evolution of the process can be predicted probabilistically.

Conformance checking this is the act of ensuring the process model ie. the result of the Process Discovery phase fits real world behaviour. In this work token-based replay is used on the workflow net to derive properties and metrics corresponding to how well that workflow net model fits the data. Information on this is then used to inform further probabilistic models or to justify searching for other models of process mining in the event of a poor fit.

Performance mining is the act of analysing an existing model and describing properties of that model. For example, examining cycle times, waiting times, and throughput times of particular activities or sequences of activities.

3.2 The Event Log

3.2.1 Formalisation

An event log L can be expressed as a sequence of activities denoted as e. The set of all possible activities is denoted Σ. The set of all possible sequences of activities (including the empty sequence ϵ) is Σ^*, where $(\cdot)^*$ is the Kleene star. We can denote T, a trace in the log to be $T \in \Sigma^*$ and the event log as $L \subseteq \Sigma^*$ and $|L| = \sum_{T \in L} |t|$

3.2.2 Examples

The event log has at minimum, cases, activities and the timestamps of those activities. Within each case a series of activities, called a Trace shows the activities and the order in which they are executed.

	case:concept:name	concept:name	time:timestamp
8569	case-9986	5	2011-10-26 08:07:49.759000+00:00
8570	case-9986	g	2011-10-26 08:08:14.773000+00:00
8571	case-9997	0	2011-10-18 07:03:12.303000+00:00
8572	case-9997	1	2011-10-18 07:04:48.732000+00:00
8573	case-9997	3	2011-10-18 07:05:12.359000+00:00
8574	case-9997	4	2011-10-18 07:05:30.196000+00:00
8575	case-9997	5	2011-10-18 07:06:01.468000+00:00
8576	case-9997	g	2011-10-18 07:06:20.547000+00:00

In this excerpt from an event log, the Case identification (`case:concept:name`) column is duplicated across a number of rows. Similarly, the `concept:name` column has the activities, denoted by lowercase letters and numbers, performed at times denoted by the timestamp.

	count
0	793
1	793
3	793
4	793
5	793
g	793

The data can be aggregated on the activities as below.

There are activities that are more common over the dataset and some that are less common.

3.2.3 Traces

Here, we introduce the notion of a trace. A trace, also known as a variant, is a collection of activities executed in order from left to right in an ordered list. For example:

case:concept:name	variant
case-10011	0,1,2,1
case-10017	0,5,1,2,1,g,2,1,2
case-10024	0,1,3,4,5,g
case-10025	0,1,3,4,5,g
case-10028	0,1,3,4,5,g,m,n,p,q
case-10059	0,1,3,4,5,g
case-10061	0,5,g,1,3,4
case-10062	0
case-10065	0,1,3,4,5,g
case-10066	0,1,3,4,5,g

This table shows that `'case-4810'` starts with `'0'` and ends with `'g'`.

Traces can have common starting and ending activities throughout the dataset. It is useful to analyse the dataset in terms of directly-follows relations - simply recording which activities follow each other and investigating the structure of processes through this lens.

3.3 Directly-Follows Graphs

3.3.1 Formally

We can define the directly follows graph of a log L as $G(L)$, a directed graph with nodes corresponding to activities $a, b, \ldots \in \Sigma$ and arcs drawn from node a to node b if $[\ldots, a, b, \ldots] \in L$.

3.3.2 Example

This is an area where an analogue between process mining and data mining can be made. A directly follows graph is to process mining as a scatter plot is to traditional data mining. Subsequent analysis often depends on the analysis of this type of structure. Below is a table showing the most frequent arcs drawn in a particular event log,representing the pairs of activities most often following each other.

	Frequency
(3, 4)	781
(1, 3)	765
(0, 1)	753
(4, 5)	713
(5, g)	713
(4, g)	80
(0, 5)	40

Here is the graph of the same data tabulated above, with line weights representing the frequency of the directly-follows relationship between two adjoining activities.

3.4 Process Trees

Process trees are a notation used to create block structured process models that can represent event logs in an abstract sense with very little loss of generality. A process tree is a tree where the leaves are

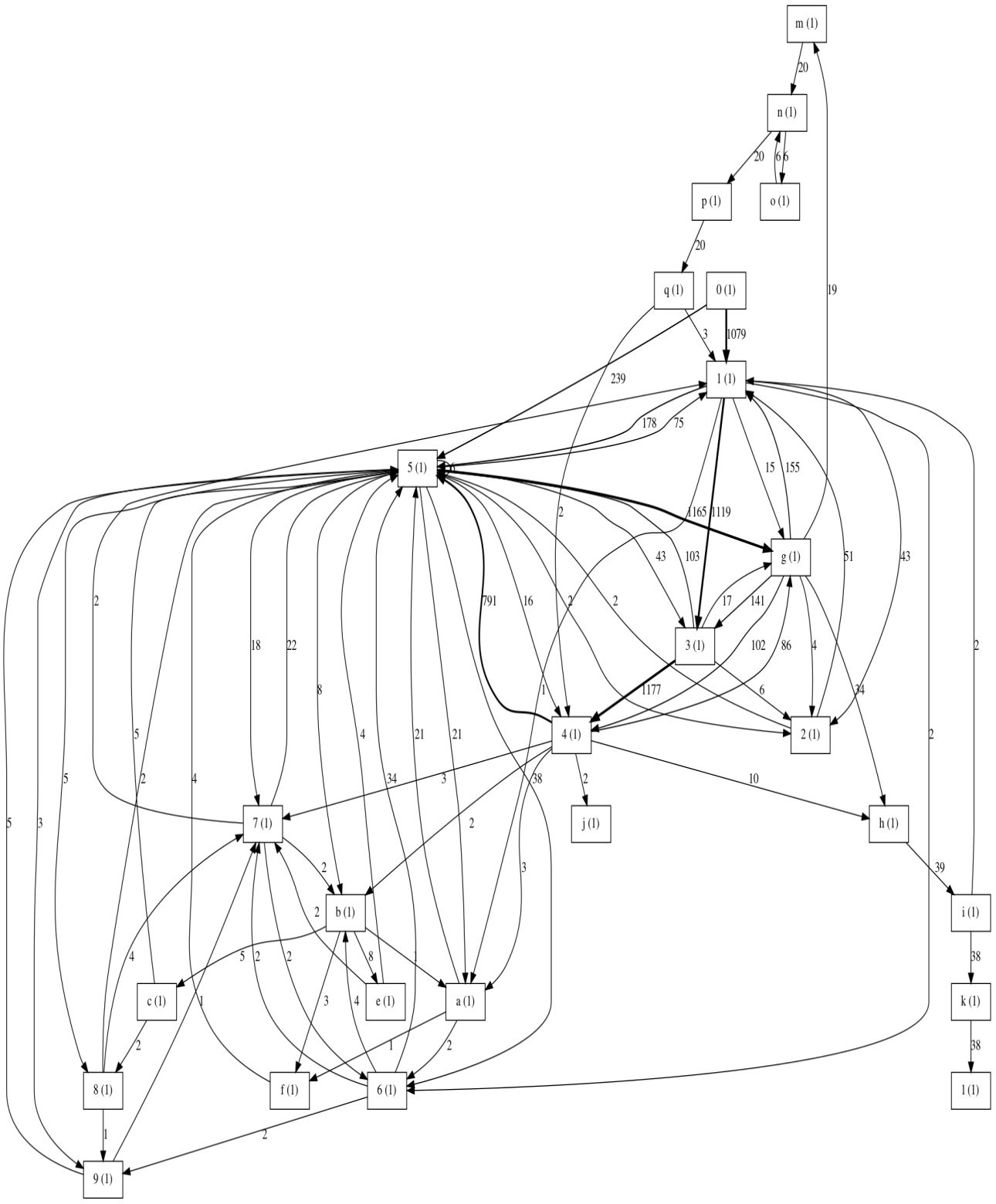


Figure 1: The Directly Follows graph of the `receipts.csv` dataset described below. Nodes correspond to activities and arcs correspond to the frequency in the log that activities that occur one after another.

individual activities, and nodes in the tree are operators on the activities. Traces can be recreated by evaluating the tree. As proven in [7] and [8], the execution of process trees results in processes that fulfil certain qualities in themselves and when translated into petri nets or markov chains later, at the cost of introducing the hidden transition τ into the activities.

3.4.1 Inductive Miner - Directly Follows Based

Directly-follows graphs often have certain characteristics that allow certain cuts to be created in the graph. These cuts map onto structures directly representable by process trees. Directly follows graphs can be cut into smaller and smaller parts, turned into process trees and concatenated onto one another. The IM framework, as detailed in [8] details 4 different operators and their characteristic structures in directly follows graphs - see fig.3 in [7].

3.5 Petri Nets

3.5.1 Informally

A petri net can be described as a directed graph with transitions and places (hence it's alternate name: place/transition graphs) with a semantics for allowing tokens to travel along those arcs.

Each unfilled circle is a place, and each labelled square is a transition. A filled circle in a place is called a token and the movement of this token is the key to understanding how these models can be used to model processes. A transition can "fire" if each of the places before it have a token in it. If a transition fires, one token is removed from each input place, and placed in an output place. For example, with N_2 above, we can keep track of the transitions that have been activated in a sequence called the firing sequence, usually denoted σ. This firing sequence will be how we construct traces in an event log. Similarly, a multiset M can be made out of the places in the Petri Net, with the multiplicity of the net representing the number of tokens in it. Denoting each place by the transition that proceeds it,(or out if it's the final place to the right) the markings go from

$$
\begin{aligned}
M_0 &= [\bullet A^1, \bullet(BC)^0, \bullet D^0, (out)^0] \quad \sigma = [] \\
&\xrightarrow{A}: \\
M_1 &= [\bullet A^0, \bullet(BC)^1, \bullet D^0, (out)^0] \quad \sigma = [A] \\
&\xrightarrow{B}: \\
M_2 &= [\bullet A^0, \bullet(BC)^0, \bullet D^1, (out)^0] \quad \sigma = [A, B] \\
&\xrightarrow{D}: \\
M_3 &= [\bullet A^0, \bullet(BC)^0, \bullet D^0, (out)^1] \quad \sigma = [A, B, D]
\end{aligned}
$$

3.5.2 Formally

a Petri net is a triple $N = (P, T, F)$ such that

- P a finite set of places
- $T \mid P \cap T = \emptyset$ a finite set of transitions
- the flow relation $F \subseteq (P \times T) \cup (T \times P)$ which dictates the "directedness" of the petri net
- then we can denote a marked petri net $N\prime = (M, N)$
- where M is a multiset $M = \mathbb{B}(P)$ showing the marking of this net.
- The set of all marked petri nets is denoted $\mathcal{N}$

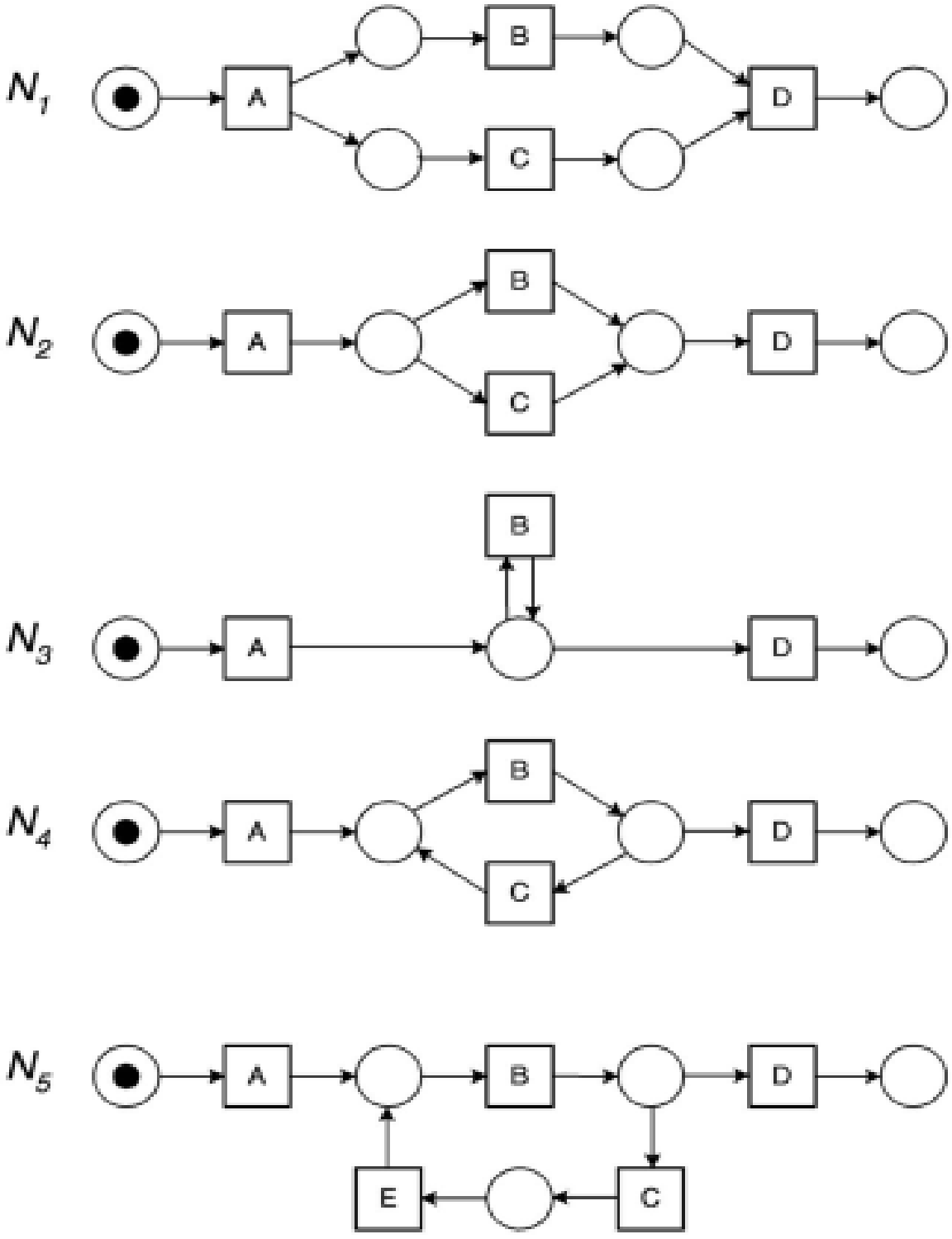


Figure 2: A set of small, sound petri nets from [7].

Defining nodes and firing rules Given a petri net N defined above

- define nodes as $P \cup T$. ie. either places or transitions are nodes.
- node x is an input node of another node y if and only if there is a directed arc from x to y (i.e $((x, y) \in F)$) denoted $\bullet x = \{y | (y, x) \in F\}$
- similarly, node x is an output node of another node y if and only if there is a directed arc from x from y (i.e $((x, y) \in F)$) denoted $x\bullet = \{y | (x, y) \in F\}$
- $\bullet a$ can be read as "nodes before a"
- and $a\bullet$ can be read as "nodes after a"

So then the firing rules themselves can be described

- let (N, M) be a marked Petri net with $N = (P, T, F),\ M \in \mathbb{B}(P)$
- Transition $t \in T$ is enabled and denoted $(N, M)[t\rangle \iff \bullet t \leq M$
- The firing rule $\_[\_\rangle \subseteq \mathcal{N} \times T \times \mathcal{N}$ is the smallest relation satisfying
- $(N, M) \in \mathcal{N} \forall t \in T\, (N, M)[t\rangle \rightarrow (N, M)[t\rangle (N, M(\setminus \bullet t \uplus t\bullet))$
- informally, the firing rule is defined as a mapping from the set of all marked petri nets to the set of all marked petri nets reachable through all transitions.
- and then it takes the markings at a M_n, takes the nodes before t, and puts them after t, and calls it M_{n+1}
- we then can construct a sequence $\sigma = [t_1, t_2, ...t_n]$ which corresponds to a set of marking multisets $\{M_i\}_{i=1}^{n}$ of the same petri net N.
- These firing sequences are how petri nets can be used to represent traces in an event log!

Firing sequences notation The notation $M \xrightarrow{t} M'$ can be used to describe using some transition t to move from marking M to marking M'.

The notation $M \xrightarrow{\sigma} M'$ can be used to describe using an specific sequence of transitions σ to move from marking M to marking M'.

The notation $M \xrightarrow{*} M'$ can be used to describe using an unspecified firing sequence transition to move from marking M to marking M'.

3.5.3 Useful properties of Workflow Nets

Ideally, we want to use Petri Nets to model a sequence of activities. However, subject to the notation above, there are restrictions on which Petri nets are actually useful to this end. These Petri nets are called Workflow nets. These nets have the following three properties.

Option to Complete A workflow net firstly has two important places

- a start place i
- st. $\bullet i = \emptyset$
- and a completion place o
- st. $o\bullet = \emptyset$

A workflow net will have a firing sequence that allows a token to end up in the final state $[o]$ given that it starts in the state $[i]$. This is guaranteed if each couple of places and transitions of the petri net (P, T) are strongly connected. [2]. Formally, $\forall M, ([i] \xrightarrow{\star} M \implies M \xrightarrow{\star} [o])$ This condition is called **option to complete**.

Proper completion A workflow net can not only be completed, it can be completed in such a way that the only marking left is $[o]$. Formally, a workflow net satisfies **Proper completion** if $\forall M, [i] \xrightarrow{\star} M \wedge M \geq [o] \implies M = [o]$

No dead transitions A dead transition is a transition that impossible to enable. A transition that is impossible to enable means that it is impossible to collect enough tokens before it to allow it to fire. Formally, $\forall t \in T, \exists M, M' : [i] \xrightarrow{\star} M \rightarrow$
M′

3.5.4 Given those three properties, what does that imply?

If all three of those properties hold, the Petri net is called a **sound** Workflow net. In addition, the existance of a single input and output node implies (see [2]) that the workflow net is **k-bounded** for some initial starting amount of tokens k. It also places a bound on the number of times a transition is featured in a net. In addition, due to constraints placed by the representational bias of process trees the Workflow nets found by the Inductive miner are well-structured, free-choice [7] These are conditions in [2] that allows k to be bounded at 1 and denoted **1-bounded** or **safe**, which result in Workflow nets that are more amenable to representation as a Transition diagram (see section BLAH). As discussed in the section above on discovering process trees and in [12], a process tree necessarily maps to a sound workflow net. While not all workflow nets map to a process tree, all process trees map to workflow nets. The use of the Inductive miner has disadvantages - namely the use of the invisible transition τ used to allow Petri net traversals that would otherwise not be possible in terms of the Petri net and the event log. This promotes a high fitness (see next section) but may adversely affect simplicity and readability.

3.5.5 An example workflow net

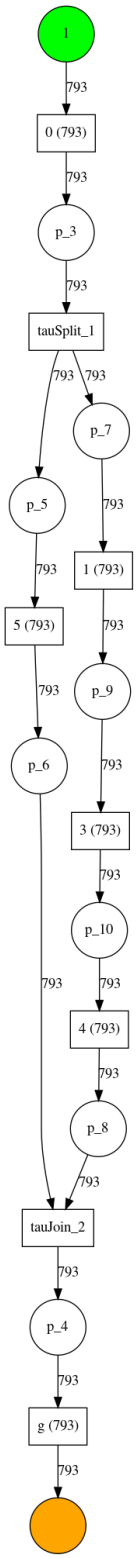


Figure 3: A workflow net discovered from the `receipts.csv` dataset. The petri net above is a workflow net.It has specific start and end places, indicated in green and gold respectively.

3.6 Petri net replay

A sense of how well the workflow net fits the data can be defined in terms of how tokens move through the petri net. This process is called token-based replay. Each one of the traces in the event log are "played into" the workflow net. The trace is reproduced by attempting to traverse the workflow net using a given trace as a firing sequence. Invisible transitions can be used to move around the petri net without adding to the firing sequence. The number of tokens consumed and produced (by activating transitions) is recorded. This is important in the context that not all process models are guaranteed to fit and that unfit models can produce tokens in unexpected places which affect the playback of the model. Notably in the case of Petri nets, four metrics are defined in terms of how many tokens are produced and consumed in the process of firing transitions and how many tokens are created and deleted for the sake of getting the petri net to complete.

In the replay, the amount of tokens produced, consumed, missing and remaining are defined as (p, c, m, r) respectively.

3.6.1 Fitness.

Fitness quantifies the extent to which the discovered model can accurately reproduce the cases recorded in the log. In a token based replay, this is calculated as

$$\text{Fitness}(m, c, r, p) = \left(1 - \frac{m}{c}\right)\left(1 - \frac{r}{p}\right)$$

This is 1 for perfectly fitting traces and $0 < \text{Fitness}(m, c, r, p) < 1$ for non-fitting traces.

3.6.2 Simplicity.

Process mined petri nets can often be a useful tool to describe a process on a human level. It is often desirable to seek the simplest model that describes the data, bearing in mind it how possible it is to produce workflow nets and petri nets described as "spaghetti diagrams". Workflow nets are not implicitly designed with readability in mind, and as such this is often a low metric for them but process discovery algorithms that strongly focusses on simplicity such as the α-miner and the SIMPLE algorithm provided by [3]. These discovery techniques generally result in small, simple models, but with low replay fitness.

$$\text{Simplicity} = \frac{1}{1 + \frac{|\text{ no-places }|}{1 \text{ no-transitions }|}}$$

3.6.3 Generalization.

Generalization assesses the extent to which the resulting model will be able to reproduce future, unspecified in the log behavior of the process. In addition, it also measures how broad the scope of the model is with respect to sequences of defined activities. For example a very precise model could be made that fits each individual case in the log as a separate path in the model. This is clear overfitting and this score seeks to diagnose cases like it.

$$\text{Generalisation} = 1 - \frac{\sum_{t \in \text{ transitions}} \sqrt{\frac{1}{\text{n.occreplay } (t)}}}{|\text{ transitions }|}$$

3.7 (Memoryless) Stochastic Petri nets

Petri nets are formalised in such a way that while the flow relation between places and transitions tells us how to get from place A to B, the transitions are simply a means to that end. The standard formalism only specifies when a transition **can** be fired, not when they **are** fired. To that end, an additional object is added to the Petri net triple $N = (P, T, F)$. Define Λ, a set of positive firing rates

associated with each transition. Formally, $\Lambda = \{\lambda_t\}_{t \in T} \in \mathbb{R} : \lambda > 0$, and a stochastic Petri Net is then $SN = (P, T, F, \Lambda)$.

These firing rates dictate how often a transition is activated per unit time. Given a transition $t \in T$ we can define the sojourn time in $\bullet t$ as a random variable $S_{\bullet t}$. This simple notion of making transitions occur randomly across all activated transitions implies that the transition firing rate is **memoryless** - implying that in the continuous time case $S_{\bullet t} \sim Exp(\lambda_{\bullet t})$ This allows the notion of completing a Workflow net to relate to timespans defined in the process and allows you to ask questions of the model like "how long does the model take to complete" without explicitly introducing timing mechanisms into the Petri net formalism. However, it does enforce a heavy-handed assumption of exponential sojourn times on any event logs analysed in this way. The implications of this assumption is discussed further through this paper.

3.8 Transition Systems and Reachability

A transition system is a simple way of modelling processes in a fashion that uses directed graphs. It is similar to a Petri net and is described in [6] as a triplet (A, S, T) of the activities, states and transitions but has key differences in it's semantics. A is a set of all activities that would appear in an event log and S is the series of states of the system. The activities however, have a one-to-many relationship with T, the transitions, which generally have are a subset of $S \times A \times S$ structure (where $(\times)$ is the cartesian product). Transitions here exist between states which can be defined for most if not all process mining techniques. While the simplicity of transition systems is useful, they have a fundamental problem with representing concurrency as discussed in [6]. If attempting to model N concurrent activities - take for example an `AND` split in a Petri net - in this situation, $|A| = N$ and there are $|M| = m$ tokens. One has to create $|S| = (m+1)^N$ states to account for that process, which can result in very large - or possibly even infinitely sized - transition graphs.

3.8.1 Reachability

The reachability graph is a way of converting a Petri net into a transition system. The states of the transition system S correspond to markings M of the Petri net. Each state is a specific marking of the petri net.

Working with workflow nets makes the transition graph much more tractable. Workflow nets are sound and bounded, meaning all transition graphs of workflow nets are bounded. Given that the inductive miner creates well-structured workflow nets, they are going to be safe ie. 1-bounded, leaving the worst case $|S| = 2^N$ for N activities all acting in parallel.

3.8.2 A reachability graph

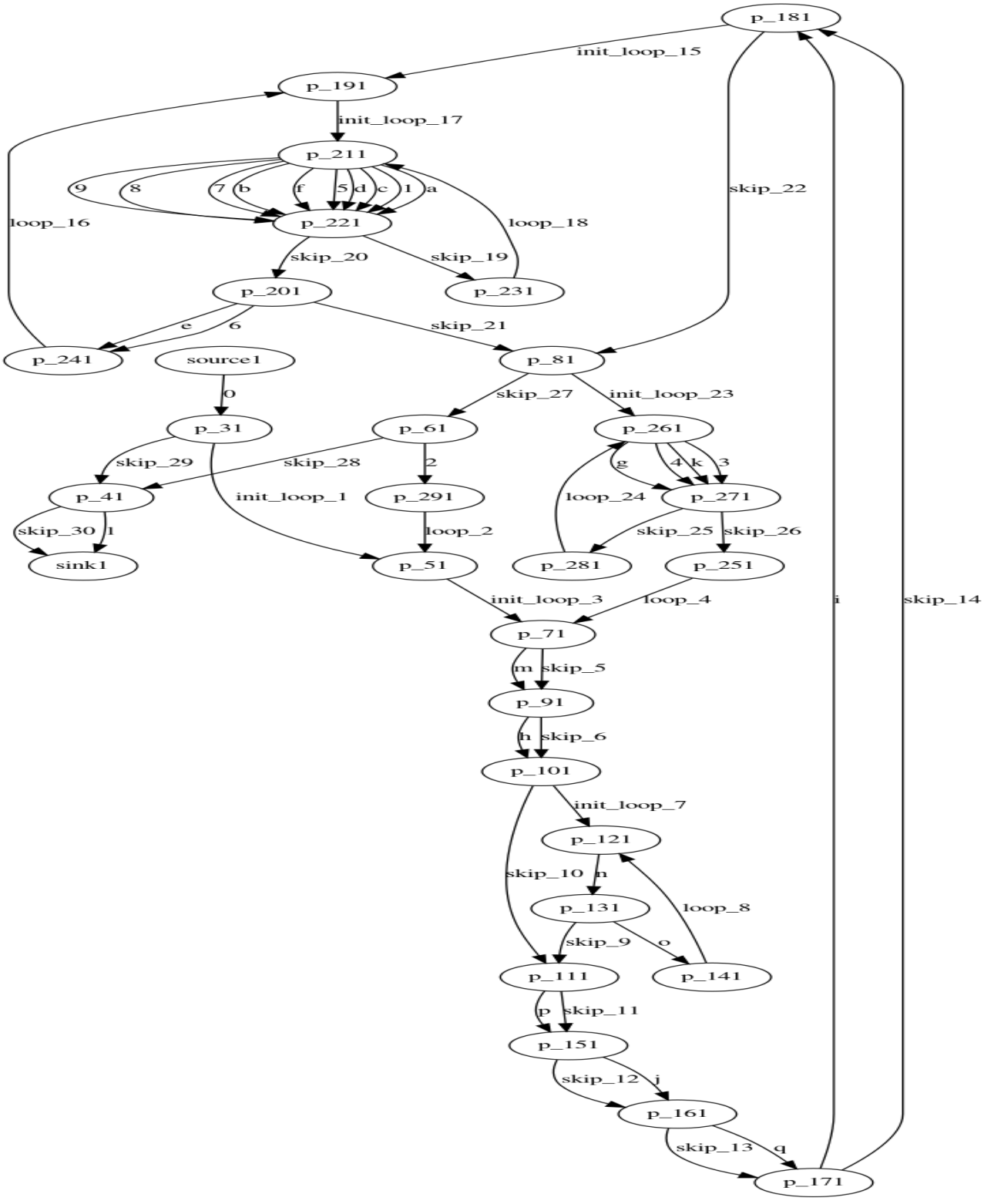


Figure 4: The reachability graph of the dataset described above. A reachabilty graph is a transition system based on the states in a petri net. Nodes correspond to permutations of possible locations of markings in a petri net for a given initial condition of $M = \{source1^1\}$

3.9 Reachability graphs from transition systems

A Stochastic Petri Net $SPN = (P, T_{pn}, F, \Lambda_{pn})$ The transition system of SPN is a tuple $TS = (A, S, T_{ts}, \Lambda_{ts})$. The state space S is bounded and as such represents a countable state space for a Markov chain. The transitions occur in SPN with a probability provided by a memoryless distribution, with parameters $\lambda_{pn,i} \in \Lambda_{pn}$. These probabilities can be found using maxiumum likelihood estimation for the parameter $\lambda_{t,pn}$ with respect to the data associated with that activity in the event log. Λ_{pn} is a bijection between elements of T_{pn} and the parameter of the memoryless distribution function $\lambda_{pn,i}$ Similarly, Λ_{ts} in the transition system is a bijection between T_{ts} and the parameters of the memoryless distribution $\lambda_{ts,i}$. While generally, $T_{ts} \neq T_{pn}$ and $\Lambda_{ts} \neq \Lambda_{pn}$, for the most their elements are broadly similar. Where a transition in a petri net only consumes and produces an equal amount of tokens, the transition between those corresponding states in a transition system is exactly the same. However, if the amount of tokens is not preserved over a transition, the transition system expands to capture the combination of tokens in the petri net. This leads to non-unique transitions in the transition system.

Another issue comes from the situation where there are multiple transitions between two places in a stochastic petri net. This would correspond to an `XOR` split in a Petri net and raise the issue of creating a mixture distribution of memoryless distributions. Given that transition time is independant of the amount of tokens in a place, and a transition is activated only when the place before it has a token in it, it only needs to fire once between all transitions in the `XOR` split for the transition to occur. Memoryless distributions have properties which allow the simplification of this problem, at the cost of discarding some information about **which** transition fired.

3.10 Markov Chains

The notion of memoryless transitions in a transition system can be represented well in a stochastic process known as a Markov Chain. A summary of some important results is presented here. Further proofs of results can be found in [13] and [4]. Broadly speaking, a Markov chain is a random process over a countable state space (eg. an interval $[-x, x] \in \mathbb{Z}$, a graph with N nodes etc) where you can assign a probability of going from one state to another for each pair of states. This transition probability is only dependant on *where you are at a given time* rather than *where you have been.*

3.10.1 Generally...

A Markov Chain is a stochastic process: a sequence of random variables $[X_t]_{t \geq 0}$ that resolve to states $S = \{x_t\}$ ith the **Markov Property** defined below.

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \ldots, X_1 = x_1, X_0 = x_0)$$

$$= P(X_t = x_t \mid X_{t-1} = x_{t-1})$$

This means the probability of being in a state at time $t = t + 1$ only depends on where you were at $t = t$, where you are at $t = t$ only depends on where you were at $t = t - 1$, and so on until $t = 0$. While Markov chains can be defined in a time-inhomogenous sense (where transition probabilities depend on the total time in the system as a whole), the Markov chains used here are all **time-homogeneous** and as such

$$\forall t \geq 0,\ P(X_{t+1} = x_{t+1} | X_t = x_t) = P(X_t = x_t | X_{t-1} = x_{t-1})$$

As such, it obeys the Chapman-Kolmogorov equations that simplify calculating transition probabilities. (see below.)

$$i, j \in S$$

$$\mathrm{P}(\mathrm{X}_{t+s} = x_{t+s} | X_t = x_t) = P(X_{t+s} = j | X_t = i)$$

$= \mathrm{P}_{i,j}^{(s)} = \sum_{k \in S} P_{i,k}(s) P_{k,j}(0)$

$= \sum_{k \in S} P_{k,i}(s) P_{j,k}(0)$

$\mathrm{P}_{i,j}^{(s+t)} =$

$= \sum_{k \in S} P_{i,k}(t+s) P_{k,j}(t)$

$= \sum_{k \in S} P_{k,i}(t+s) P_{j,k}(t) =$ Chapman Kolmogorov equationGiven that there are $|S| = N$ states in the chain, the state vector is a $N \times 1$ vector denoting what state the system is in at time t,x_t, $x_t \mathbf{P}^s = x_{t+s}$, allowing us to interpret P^s (the transition matrix exponentiated by s) as a time evolution operation.

3.10.2 Properties of paths

It is possible to find the total probability of going from a state i to a state j by taking the product of one step probabilities of each intermediate state. Denoting the path through the chain as a sequence states of length m, $[\sigma]_{k=0}^{k=m}$ such that $\sigma_0 = i, \sigma_m = j$ to be

$$P_\sigma \quad = \prod_{k=1}^{k=m} [P_{k-1,k}]$$

by the Chapman-Kolmogorov equation. Given a trace $\sigma = [a, b, c, d, e]$, the probability of taking this path from $\bullet a$ to $e\bullet$ is

$$P_{i,j} \quad = P_{i,a} \cdot P_{a,b} \cdot P_{b,c} \cdot P_{c,d} \cdot P_{d,e} \cdot P_{e,j}$$

It is possible to go further with this. If, for example, a sojourn time is associated with each probability (as it will be in the continuous case) this expression can be used to find the expected travel times associated with each path through the chain.

$$\begin{aligned} E_\sigma[S_{i,j}] \quad &= \sum_{k=1}^{k=m} E[S_{k-1,k}] \\ &= \sum_{k=1}^{k=m} S_{k-1,k} \cdot P_{k-1,k} \end{aligned}$$

This is a result that will be used later to determine if a path through a process is memoryless.

Properties $P_{i,j}^{(s)}$ is a $|S|x|S|$ square matrix that can be read as "the probabilty of getting from state i to state j in s time steps." The dynamics of the chain can be derived from the rows of the matrix.

Hitting time the hitting time of a is the first time it takes for a Markov chain to be in this state a from an initial state i. $k_{i,a} = \underset{t}{argmin}[t \geq 1 : X_t = a | X_0 = i]$

Transient state A **transient** state is a state for which $P(T_{i,i} < \infty | X_0 = i) < 1$

Recurring state A **recurring** state is a state that is not transient. $P(T_{i,i} < \infty | X_0 = i) = 1$

Absorbing state A state $A \in S$ can be called **absorbing** if $P_{A,A} = 1$. The chain is called an **absorbing chain** if it is possible to reach A for each $i \in S$.

all absorbing states are recurring.

A Markov chain with transition matrix $\mathbf{P}$ absorbing chain with $|A| = a$ states with r transient states can be decomposed into a block matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

where

- $\mathbf{Q}$ a square $(a-r) \times (a-r)$ matrix of transient states
- $\mathbf{R}$ a $r \times (a-r)$ matrix showing the probabilities of transitioning into absorbing states from transient states
- $\mathbf{0}$ is the $r \times (a-r)$ zero matrix showing the probabilities of transitioning from an absorbing state into a recurring state - ie. 0 for all states.
- $\mathbf{I}$ is the $r \times r$ identity matrix.

This decomposition of the transition matrix is useful in solving for hitting probabilities and times.

3.10.3 Discrete time

Formally The Chapman-Kolmogorov equations imply that a probability from each state to each state for some unit time can be defined as:

$$\begin{aligned} \forall i, j \in S \\ 0 < s < t \in \mathbb{N} P(X_{t+s} = j | X_t = i) = P_{i,j}^{(s)} \\ P(X_{t+1} = j | X_t = i) = P_{i,j}^{(1)} = P_{i,j} \end{aligned}$$

Further properties include

Row-stochasticity $\sum_{j=1}^{|S|} P_{i,j} = 1$

Stationary distribution A stationary distribution, denoted π for the transition matrix $\mathbf{P}$ can be defined in terms of a left eigenvalue problem - $\pi\mathbf{P} = \pi\mathbf{1}$. By solving for the eigenvectors of the system where the eigenvalues of $\mathbf{P}$ are 1, the corresponding eigenvectors are $C\pi$ where C is some normalising factor that can be removed by forcing the additional constraint $\sum_i \pi_i = 1$

The fundamental matrix The fundamental matrix $N_{i,j}$ of a markov chain is the matrix that describes the expected amount of times that the chain will transition from state i to state j. Noting that for a given power of $\lim_{s \to \infty} Q_{i,j}^s \to 0$, we could find some meaning in what an infinite sum of Qs would look like. As $|Q| < 1$, this means that the infinite sum $\sum_s^{\infty} Q^s = (I - Q)^{-1} = N$. The result of this summation is the fundamental matrix.

From this, we can find the probability of absorption $P(X_t = a) = NR$ where R is defined in the block matrix above. We can also find the expected amount of jumps from state i until absorption $E[T_{i,a}] = N \cdot \mathbf{1}$.

3.10.4 Continuous Time Markov Chains

Note: as with the section above, the majority of this section is concerned primarily with the presentation of some useful results rather than detailed proofs. Further information can be found in [14] .

Formal Construction Another way of arriving at the notion of a transition matrix is by using infinitesimal times in the Chapman-Kolmogorov equation given the notion of a memoryless process.

$$\begin{aligned} \forall \quad & t \geq 0, i, j \in S \\ \text{as} , \quad & \lim_{h \to 0} \\ & P(X_{t+h} = j | X_t = i) = \delta_{i,j} + q_{i,j} h + o(h) \end{aligned}$$

- $\delta_{i,j}$ is the Kronecker delta
- and $q_{i,j}$ is a parameter parameterising the transition from state i to j.

A note on exponentially distributed random variables If a random variable X can be said to be exponentially distributed (ie. $X \sim Exp(\lambda)$) it has a few interesting properties.

1. It is defined only for $t \geq 0$
2. Probability density function :: $X \sim Exp(t) = \lambda e^{-\lambda t}$
3. Cumulative probability function :: $P(X \leq t) = 1 - e^{-\lambda t}$
4. Survival function (complement cumulative probability function) :: $P(X > t) = e^{-\lambda t}$
5. Memorylessness :: $P(X > t + s \mid X > t) = P(X > s)$. Note: this is the **only** continuous memoryless distribution.

How to test exponential-ness Exponentiality is tested in three ways for the sake of this dataset.

1. Graphically
2. Explained Variance (R^2)
3. Anderston Darling test (A^2)

The first property is assessed by looking at the histograms of the transition times and comparing them to the corresponding density of the exponential distribution it's fit to. Comparing the two visually may be enough to rule out the most egregious misfits but not all. As such, most of the testing is done with the CDF of the fit distribution against the ECDF. The ECDF is known as the Empirical Cumulative Distribution Function and is defined as

$$\begin{aligned} & \text{Given data} \{X_i\}_{i=1}^n \\ & \hat{F}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[X_i \leq x]] \end{aligned}$$

This is a good estimator for the CDF [13]. Both tests here are defined in terms of the difference $\hat{F}_n(x) - F(x)$. As there is only one parameter λ_i for each transition fit, R^2 is an acceptable indicator of fit, where R^2 can be defined in this context as

$$\begin{aligned} \sigma^2_{res} = MSE(F(x), \hat{F}(x)) = \left(\frac{1}{n-1} \right) (\hat{F}_n(x) - F(x))^2 \\ \sigma^2 = Var[\hat{F}(x)] \\ R^2 = 1 - \frac{\sigma^2_{res}}{\sigma^2} \end{aligned}$$

Finally, the Anderson-Darling test is used on the transition times. The Anderson-Darling is a variation of the Kolmogorov-Smirnov test that weighs the evidence of a sample coming from a given

distribution. The null hypothesis applied to each set of data corresponding to the transition times of each activity is: [15]

$$H_0 : S_i \sim Exp(\lambda_i)$$

A test statistic can be calculated as follows. If the test statistic is greater than the tabulated values, the null hypothesis is rejected.

$$\text{Given data}\{X_i\}_{i=1}^{n}$$
$$A^2 = -n - \sum_{i=1}^{n} \frac{2i-1}{n} \left[\ln\left(F\left(X_i\right)\right) + \ln\left(1 - F\left(X_{n+1-i}\right)\right)\right]$$

This test is most sensitive to deviations from the fit distribution around the tails, but is generally useful for all parts of the distribution.

Properties

Memorylessness implies exponential sojourn times Consider $X_0 = i$, and $X_{S_i} = j$, where S_i is the first leaving (sojourn) time from i.

$$\begin{aligned} &P(S_i > t + s \mid S_i > t) \\ &= P(X_{t+s} = i \mid X_t = i) \text{by markov property} \\ &= P(X_s = i \mid X_0 = i) \text{by time homogeneity} \\ &= P(X_s = i) \text{... but wait! thats } P(S_i > s) \\ &\implies \mathrm{P(X > t + s \mid } X > t) = P(X > s). \end{aligned} \tag{2}$$

The requirement that the sojourn times are memoryless AND continuous implies that $S_i \sim Exp(\lambda_i)$

The transition rate matrix A transition matrix with specific probabilities can only be defined for use over time intervals, rather than intervals of infinitesimal time. Constructing a $P_{i,j}$ in a similar fashion as above with the discrete case results in having entries like $P_{i,j}^{(h)}$, which would not be very useful in the limit where $h \to 0$. It does have its uses when used in terms of sojourn times (see the section below on the jump chain construction) as in $P_{i,j}^{(S_i)}$, it is not as fundamental a structure as it is in the discrete time case. Given $S_i \sim Exp(\lambda_i)$...

$$\begin{aligned} \lim_{h\to 0} P(S_i < h) &= 1 - e^{\lambda_i h} \\ &\approx 1 - (1 - \lambda_i h) \text{by taylor approximation for very small h} \\ &= \lambda_i h \\ &\text{rearranging for } \lambda_i \\ \lambda_i &= \lim_{h\to 0} \frac{P(S_i<h)}{h} \\ &= \lim_{h\to 0} \frac{P(X_h=j|X_0=i)}{h} \end{aligned}$$

We can further split this up into rows of a transition rate matrix. Denoting $p_{i,j}$ as the jump probabilities $P^{S_i}_{i,j}$, we can define the transition rate matrix as the transition rate matrix $\mathbf{Q} = q_{i,j}$.

$$\begin{aligned}
&\text{for } i \neq j \\
&\qquad q_{i,j} \qquad = p_{i,j}\lambda_i \\
&\text{for } i = j \\
&\qquad q_{i,i} \quad = -\textstyle\sum_{j\neq i} q_{ij} \\
&\qquad\qquad = -\textstyle\sum_{j\neq i} \lambda_i p_{ij} \\
&\qquad\qquad = -\lambda_i \textstyle\sum_{j\neq i} p_{ij} \\
&\qquad\qquad = -\lambda_i
\end{aligned}$$

The Jump Chain For every continuous time markov chain, there is an underlying discrete time equivalent. This is known as the embedded markov chain or the jump chain. Given a CTMC as a stochastic process: a sequence of random variables $[X_t]_{t\geq 0}$ that resolve to states $S = \{x_t\}$, The embedded chain is $Y_n = \{X_{S_i}\}$ where S_i is the sojourn time from state i. It has a transition matrix $p_{i,j} = P(X_{S_i} = j \mid X_0 = i)$ - a conditional probability that says "given that the chain leaves state i at S_i, where will it go?"

This can be found by

$$p_{ij} = \begin{cases} \frac{q_{i,j}}{\sum_{k\neq i} q_{i,k}} & i \neq j \\ 0 & i = j \end{cases}$$

Kolmogorov Forward and Backward equation Two differential equations can be used to represent the time evolution of this chain. Given a vector of $|S|$ probabilities at time t states $P(t)$,

$$\begin{aligned}
\textbf{Kolmogorov Backward Equation} \qquad & \tfrac{dP(t)}{dt} = Q\,P(t) \\
\textbf{Kolmogorov Forward Equation} \qquad & \tfrac{dP(t)}{dt} = P(t)\,Q
\end{aligned}$$

This can be solved by noting that it is a homogeneous first order linear ordinary differential equation with a solution in terms of the matrix exponential

$$P(t) = \exp[-At]P(0)$$

Hitting times and solutions of equations involving Q A number of useful systems of equations can be found in terms of the fundamental matrix The fundamental matrix can similarly be defined for the embedded jump chain and used to determine the probabilities of absorption and average number of jumps until absorption. The transition rate matrix can also be used in a number of ways to find properties of the chain.

Stationary distribution Using the Kolmogorov equation, it is possible to consider the system of equations $QP(t)$ as the amount that $P(t)$ will change over an infinitesimal amount of time dt, thus finding $\frac{dP}{dt}$. On that note, a system with $QP(t) = 0$ can be interpreted as a distribution over states that does not change over time. Similarly to the discrete time case, a distribution over states π such that $\pi Q = 0$ can be solved using linear algebra techniques. This can be considered the long-run behaviour of the chain as $t \to \infty$. For absorbing chains, this will generally be a vector of 0s with a 1 in the absorbing state.

Hitting times The hitting probability of a state $P(X_{T_A} \in A \mid X_0 = i) = h^A_i$ is defined as the probability of getting to a state in a $A \subset S$ from i through all possible routes in the chain. Using some

initial conditions, it is possible to deduce a system of equations that will represent this system. Given that $h_A^A = 1$ - stating that the probability of going to a state in A from A is 1...

$$\begin{aligned} h_i^A &= 1 \quad i \in A \\ \sum_{i \notin A} q_{i,j} h_j^A &= 0 \quad i \notin A \\ -q_{i,i} h_i^A + \textstyle\sum_{i \neq j} q_{i,j} h_j^A &= 0 \end{aligned}$$

Matrix Form

$$Q_{i,j} h_j^A = \mathbf{1}[i \in A]$$

Average hitting times A similar system can be constructed for hitting times if the sojourn times can be taken into account. The hitting time of a state $E[T_A] = k_i^A$ is defined as the mean time taken through all paths of the chain of getting to a state in a $A \subset S$ from i. Given that $k_A^A = 0$ - stating that the time taken going to a state in A from A is 0...

$$\begin{aligned} k_i^A &= 0 \quad i \in A \\ q_{i,i} k_i^A - \textstyle\sum_{i \neq j} q_{i,j} k_j^A &= 1 \end{aligned}$$

Matrix Form

$$-Q_{i,j} k_j^A = \mathbf{1}[i \notin A]$$

4 Method and Model Implementation

4.1 PM4PY

The analysis below was performed using `PM4PY 1.1.23` [3] , `scipy 1.2.1` [11] , `numpy 1.16.2`,and `pandas 0.24.2` [10] on `Python 3.7`, . Python and pandas were used to manipulate, preprocess and tabulate the data while process mining.

PM4PY was used to perform the bulk of the process mining specific operations.

- manipulating data on a per-trace basis(sampling, filtering by occurance)
- implementing and fitting directly-follows graphs, petri nets and process trees
- conformance checking, calculations of fitness, precision and generalisation and calculation of event frequency and duration through token-based replay
- generation of transition graphs from petri nets

`scipy` and `numpy` were used where necessary to create and manipulate matrices, as well as for linear algebra routines used to solve systems of equations and investigate linear algebraic properties of those matrices.

4.2 Discovering a suitable workflow net

The discovery of a workflow net looks like this

[Event Log] → [Directly Follows Graph] → [Process Tree] → [Petri Net $(sound, safe)$]

However, before doing so, a few caveats on using `PM4PY`'s `IMDFb` algorithm must be heeded. The `IMDFb` algorithm is based on the IM_D miner presented in [8]

4.2.1 Inductive miner and noise sensitivity

The inductive miner uses every trace in the event log, regardless of its completion status, start activity, finishing activity, and outlying properties of any positioning of activities in a trace or properties of the activities themselves. A workflow net discovered by the `IMDFb` miner will use invisible transitions τ to allow activities to be skipped and looped through. Infrequent behaviour will be accounted for in the model by adding a number of invisible transitions to the workflow net.

For example, in an event log like $[a, b, c, d, e]^{10}, [a, b, c, e]$ the `IMDFb` miner will account for $[a, b, c, e]$ by creating an invisible transition allowing $c\bullet$ to be transitioned out of without letting d fire. That is, $\bullet(\bullet e) = \{d, \tau\}$ A small error in data collection can cause many variants in the data like above to form and create many transition skipping τ transitions. This is generally not a problem but for noisy datasets, logical "structures" in the noise may end up creating erronious places that make sense in the context of the model but create extra complexity on the back of noise.

Another example would include $[a, b, c, d, e]^{10}, [b, c, d, e]$

For example, $[a, b, c, d, e]^{10}, [a, b, c, e], [a, b, d, e], [a, b, e]$ may be accounted for by creating an inclusive `OR` block over the transitions c, d, τ. As the end goal of creating this workflow net is to investigate its time evolution with the transition graph, which scales exponentially with the number of places in the petri net, ways of representing the process that keep places in the petri net to a minimum are preferred. Many of these invisible transitions will rarely be used but could have a large effect on the model as a whole. At the same time, fitting to as much observed behaviour as possible is desirable as it allows the workflow net to include a greater amount of behaviour.

While care was taken to identify problematic infrequent behaviour, incomplete and empty traces, the fitness of the model would be effected with respect to the unfiltered data. A fit workflow net cannot account for out-of-sample activities, and so a balance must be struck between removing problematic traces and preserving model fitness. This is made quantifiable by the metrics defined in `PM4PY`.

4.2.2 Preprocessing the data appropriately

Invalid traces and activities In the context of process mining, there may be many different definitions of an invalid trace. For example, there may be extraneous unrelated activities in an event log that may be of interest to another investigation. To use the analogy in section (cake), if the aim is to create a cookbook, not every activity performed by a baker (eg. washing their hands, going on lunchbreaks, buying ingredients) may not be relevant to the task at hand.

In general, this is true of real-life event logs where a massive amount of activity may be collected by middleware or logging software, but only a fraction of that activity is meaningful to the task at hand.

In the situation where spaghetti models (see relevant chapter in [6]) are formed, insight could be gleaned from clustering traces on metadata such as who performed the process (see social network analysis available in [3]) or even using the block structure of workflow nets and analysing smaller blocks in larger workflow nets to gain understanding of a particular part of a process. In addition, in the situation where a statistical model will be associated with each transition, it could be sensible to only create a process model using only activities that can be adequately explained in terms of its assumed distribution. For example, given that a Continuous Time Markov Chain is being used to describe the time evolution of the pricess, it could make sense to only fit a process model to activities that pass some standard of fit, like the Anderson-Darling test described above.

As such, filtering individual activities may simplify model creation significantly.

Infrequent Variants As can be seen in (variants section), many traces may only occur once or twice in a log and contain slight differences between their behaviour and the behaviour of the rest of the model. Removing these traces will often remove a number of skip transitions at the cost of the fitness of the model.

Decreasing Factor method Another way to limit the effect of noisy traces on the process model would be to limit the permitted starting and ending activities that define a valid trace. By removing

traces that do not have a valid, named starting point or ending point, you could remove structures in a workflow net that "go nowhere". In constrast, keeping traces with specific ending and starting activities may allow the process model in particular to hone in on particular "paths" through a petri net. This kind of filtering is supported in `PM4PY` [3] by allowing a "decreasing factor" to be defined. For example, take a log $[a,b,c,d,e]^{100}, [b,d,c,e]^{50}, [c,d,e]^{20}, [d,e]^{2}$. The starting activites of this log occur with the following frequencies.

	a	b	c	d
Freq.	100	50	20	2
Decreasing Factor	1	0.5	0.4	0.1

Given a sequence of starting activities $s_i = [f(a), f(b), f(c), f(d)]$ sorted by frequency, the decreasing factor is defined as 1 for the most common starting activity. For each most starting activity beyond that, it is defined as $D = \frac{f(s_{i+1})}{f_s}$ The activities with decreasing factors above a specified $D^* > D$ are kept. For example, $D^* = 1$ will only accept traces beginning with a, while $D^* = 0.5$ will accept traces beginning with a or b. Similarly, $D^* = 0.2$ will accept a, b or c. A decreasing factor of $D^* = 0$ will accept all traces.

4.3 Inspecting transition distributions

4.3.1 Maximum Likelihood estimation of an exponential parameter

The transitions of the workflow net were assigned transition rates through into two categories

Visible Transitions Data was available for the amount of times over an event log this transition was activated as well as the amount of time it took for this transition to occur. The maximum likelihood estimator is the parameter $\hat{\theta}$ such that

$$\hat{\theta}^* = \underset{\hat{\theta}}{argmax} = l(\hat{\theta} \mid X_i) = \left(\frac{1}{n}\right)(\sum_{i=1}^{n}) \ln[f(x_i \mid \hat{\theta})]$$

l here is known as the log-likelihood and is a way of parameterising a probability density function in a way that is amenable to optimisation. The maximised log-likelihood can be described as the "function with parameter $\hat{\theta}$ that is the most likely given the data" ie. the value for $\hat{\theta}$ in the expression above that is **most likely** for all possible θ.

The maximum likelihood estimator for a given exponential distribution has a closed form expression related to the sample mean of the data.

$$\hat{\lambda} = \frac{1}{\overline{x}} = \frac{n}{\sum_i x_i}$$

Because of the similarity of this expression to the sample mean, much of the stochastic properties of this estimator can be analysed in the same way as a sample mean. The asymptotic properties of the sample mean follow a central limit theorem, can be subject to t-tests and have confidence intervals of it's distributions. These maximum likelihood estimators were implemented using [11].

Invisible transitions Invisible transitions are accounted for differently than in `PM4PY` CTMC methods which ignore them completely. A transition system is created with a truncated Petri net without any invisible transitions using the `SIMPLE` algorithm able to capture most information. A different approach was taken in this work - accounting for invisible transitions by assigning them a sufficiently low value of $\lambda_\tau << \min_{\lambda_i}[\lambda_i]$. This will be weighted further by accounting for the by finding the amount of times this invisible transition is used in the token-based replay to find transition probabilities.

4.3.2 Probabilities of transition from the token based replay

The token based replay capabilities in `PM4PY` allow for the probabilities of using a particular transition from the data. Collected from these replays is the amount of times a place has a token in it. Every

time a token is placed in a transition, a set of transitions are activated and ready to fire. A token is removed from a place as a transition after it fires. Given the token based replay only replays traces that are seen in the data, we can assume that the ratio of times a transition is fired to the amount of times it **could be fired** ie. activated is a good estimator of probability of an activity occuring in a process. ie.

$$P_{i,j}^{S_i} \approx \frac{\#(\text{transitions over } t)}{\#(\text{activations of } t)}$$

4.4 Turning a safe workflow net with invisible transitions into a Markov Chain

Given the elements of $\lambda_i \in \Lambda$, and the probabilities of transition $P_{i,j}^{S_i}$ from the token based replay it is now possible to construct a transition rate matrix from the workflow net. But first, the transition system must be constructed by assigning a state to each combination of possible tokens in the workflow net [5] . The transitions in the transition system are allowed be duplicates of each other, and occur with the same probability as transitions in the stochastic petri net. Where multiple transitions lead from the same place to the other the workflow net to another, the following result is used to aggregate the transitions.

$$\begin{aligned} P\left(\min\left\{X_1,\ldots,X_n\right\} > x\right) &= P\left(X_1 > x,\ldots,X_n > x\right) \\ &= \textstyle\prod_{i=1}^{n} P\left(X_i > x\right) \\ &= \textstyle\prod_{i=1}^{n} \exp\left(-x\lambda_i\right) = \exp\left(-x\sum_{i=1}^{n}\lambda_i\right) \end{aligned}$$

$$\implies \left(\min\left\{X_1,\ldots,X_n\right\} > x\right) \sim Exp\left(\sum_i \lambda_i\right)$$

The matrix Q as defined in the CTMC section can be now constructed as

$$\begin{aligned} &\text{for } \ i \neq j \\ &\qquad q_{i,j} \quad = p_{i,j}\lambda_i \\ &\text{for } \ i = j \\ &\qquad q_{i,i} \quad = -\textstyle\sum_{j\neq i} q_{ij} \\ &\qquad\qquad = -\textstyle\sum_{j\neq i} \lambda_i p_{ij} \\ &\qquad\qquad = -\lambda_i \textstyle\sum_{j\neq i} p_{ij} \\ &\qquad\qquad = -\lambda_i \end{aligned}$$

$q_{i,i}$ can now be described as $\lambda_{\bullet t}$ for a given marking involving $\bullet t$ in the workflow net. The transition graph will have a corresponding set of nodes in it for the situation where a marking is placed in $\bullet t$, and will move away from it at a rate equal to the sum of the rates of the transitions heading out of the place. $q_{i,j}$ is then the rate at which markings in a petri net as described by the states in the transition graph change from marking i to marking j.

4.5 Time evolution

Finding Q opens up the possibility of finding everything defined in terms of Q defined above. The notion of "closeness to completion" can be solved by solving the hitting time system of equations defined above and visualised with a graph showing the mean time until completion. Then by evaluating solving the Kolmogorov Forward equation using the matrix exponential solution at different times, the time evolution of the system can be visualised. The diagram below shows an example of this time evolution. The blue areas are parts of the probability distribution with a low probability at that time, and the pink areas are parts of the probability distribution with a high probability at that time.

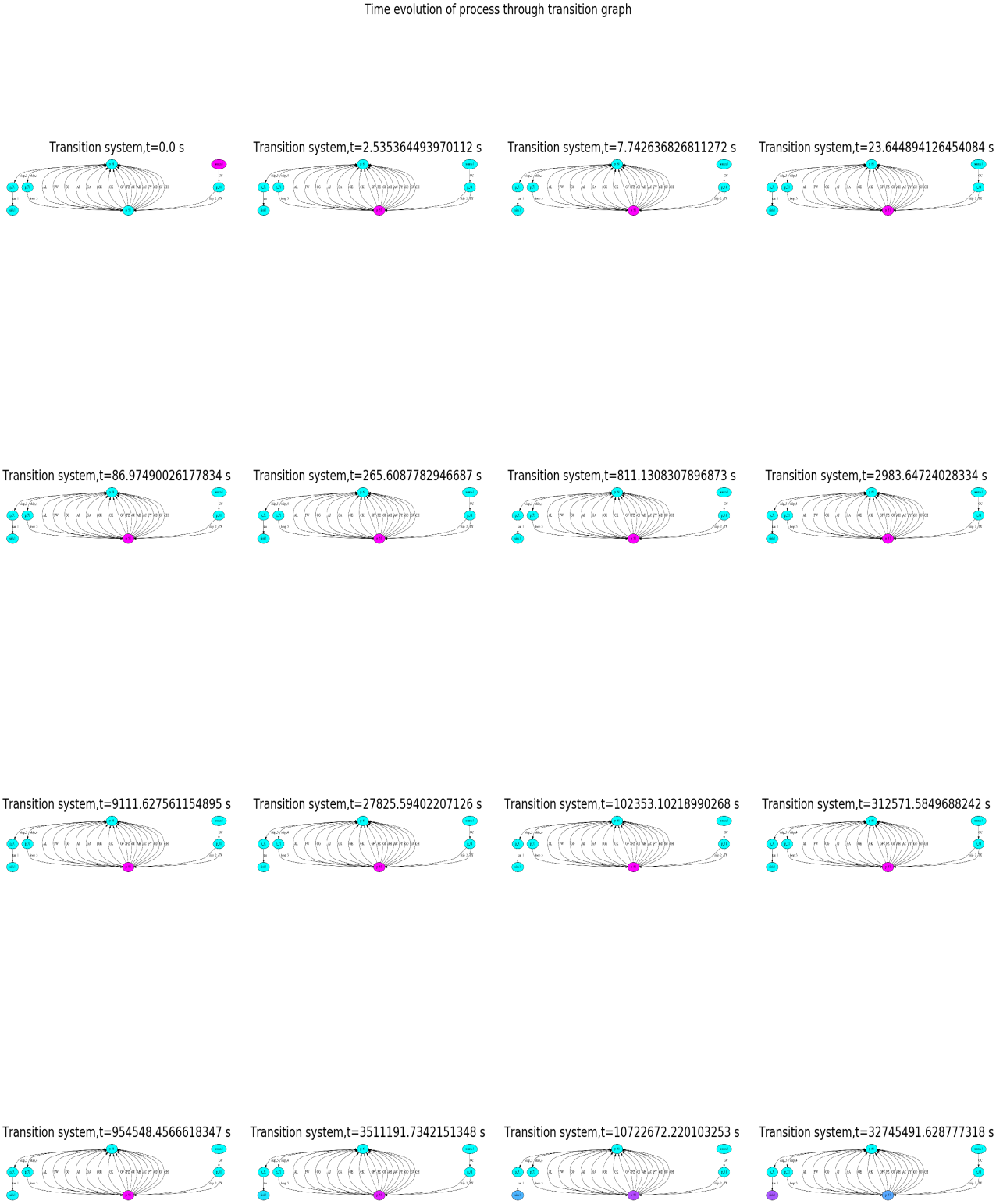


Figure 5: A reachability graph coloured by the probability of being in each state at time t ie. $P(t)$ Cyan states have probability 0, and pink states have probabilities close to 1 at the times shown.

To get a better notion of "closeness", the mean time through the transition system from each state is shown below. Times closer to 0 are coloured blue and higher times are more red.

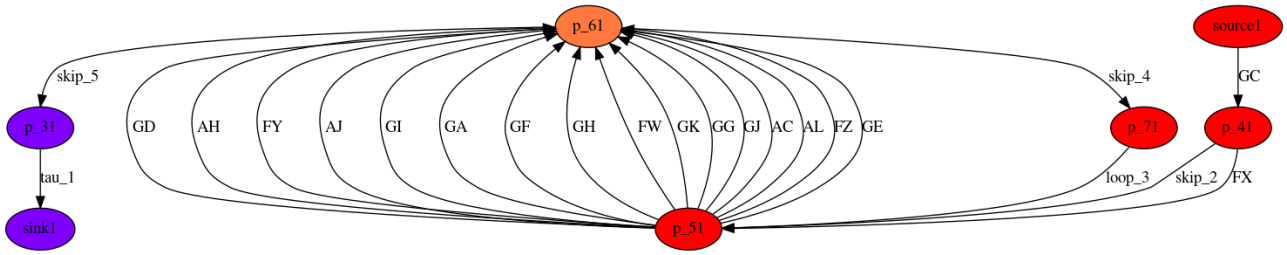


Figure 6: A reachability graph coloured by the mean hitting time it would take to get from this state to the end state `sink1`. Denoted $E_i[T_A] = k_i, A$ in the theory section, the more blue areas are close to 0 and the more red areas are close to the maximum of $E_i[T_A]$.

4.6 Variant analysis

Where it exists and is not infinite, $\frac{1}{q_{i,j}}$ can be interpreted as a cost matrix by considering the expectation of an exponential distribution $E[Exp(\lambda_i)] = \frac{1}{\lambda_i}$. This can be used to find the expected time to traverse a particular path in the transition graph, in comparison to solving a system of equations to find an average hitting time over all paths between i and $j = E_i[T_j]$. Denoting the average time to transition from a state i to state j as $S_{i,j} = \frac{1}{q_{i,j}}$, we can consider the mean time along a path a sequence of length m, $[v]_{k=0}^{k=m}$ such that $v_0 = i, v_m = j$ to be

$$E[S_{i,j}] \quad = E\left[\sum_{k=1}^{k=m} S_{k-1,k}\right]$$

by the Chapman-Kolmogorov equation.

Given a trace $\sigma = [a, b, c, d, e]$, the mean time taken to transition from $\bullet a$ to $e\bullet$ is

$$\begin{aligned} E_\sigma[S_{a,e}] \quad & = E[S_{a,b} + S_{b,c} + S_{c,d} + S_{d,e}] \\ & = E[S_{a,b}] + E[S_{b,c}] + E[S_{c,d}] + E[S_{d,e}]] \text{ by linearity of} E[\cdot] \end{aligned} \tag{4}$$

These means not agreeing would indicate that the Chapman-Kolmogorov equation does **not** hold, indicating that the memorylessness property does not hold. Welch's test (t-test for populations with unequal variances) is used to determine the null hypothesis for each variant where $E_\sigma[S_{i,j}]$ is the expected time taken to go from

$$H_0 : E_\sigma[S_{i,j}] \quad = \sum_{k=1}^{k=m} E[S_{k-1,k}] \quad = \sum_{k=1}^{k=m} \frac{1}{q_{k-1,k}}$$

In the data, $E_\sigma[S_{i,j}]$ would be the mean of the times to complete the process through the variant σ, and the sum would be the sum of the fit activity times. If these are not equal, it means the Chapman-Kolmogorov equation does not hold, which indicates that the conditions in which it holds (ie. the assumption that the process is a time-homogeneous memoryless stochastic process) are not holding as they should.

5 Results

5.1 Receipts processing

This data file is provided as an example data file provided by `PM4PY`. Each transition is denoted by an alphanumeric character and represents an activity in the process.

	concept:name	concept:name:old
0	0	Confirmation of receipt
1	1	T02 Check confirmation of receipt
2	2	T03 Adjust confirmation of receipt
5	5	T06 Determine necessity of stop advice
9	g	T10 Determine necessity to stop indication
15	3	T04 Determine confirmation of receipt
16	4	T05 Print and send confirmation of receipt
31	m	T16 Report reasons to hold request
32	n	T17 Check report Y to stop indication
33	p	T19 Determine report Y to stop indication
34	q	T20 Print report Y to stop indication
90	h	T11 Create document X request unlicensed
91	i	T12 Check document X request unlicensed
92	k	T14 Determine document X request unlicensed
93	l	T15 Print document X request unlicensed
105	6	T07-1 Draft intern advice aspect 1
106	b	T08 Draft and send request for advice
107	c	T09-1 Process or receive external advice from ...
1173	a	T07-5 Draft intern advice aspect 5
1189	f	T09-4 Process or receive external advice from ...
1216	8	T07-3 Draft intern advice hold for aspect 3
1237	9	T07-4 Draft internal advice to hold for type 4
1238	7	T07-2 Draft intern advice aspect 2
1258	j	T13 Adjust document X request unlicensed
2283	e	T09-3 Process or receive external advice from ...
4266	o	T18 Adjust report Y to stop indicition
6307	d	T09-2 Process or receive external advice from ...

There are a number of ways to slice this dataset in order to produce a workflow net. Using the methods outlined above.

Fitting on all variants, Decreasing Factor = 0.5

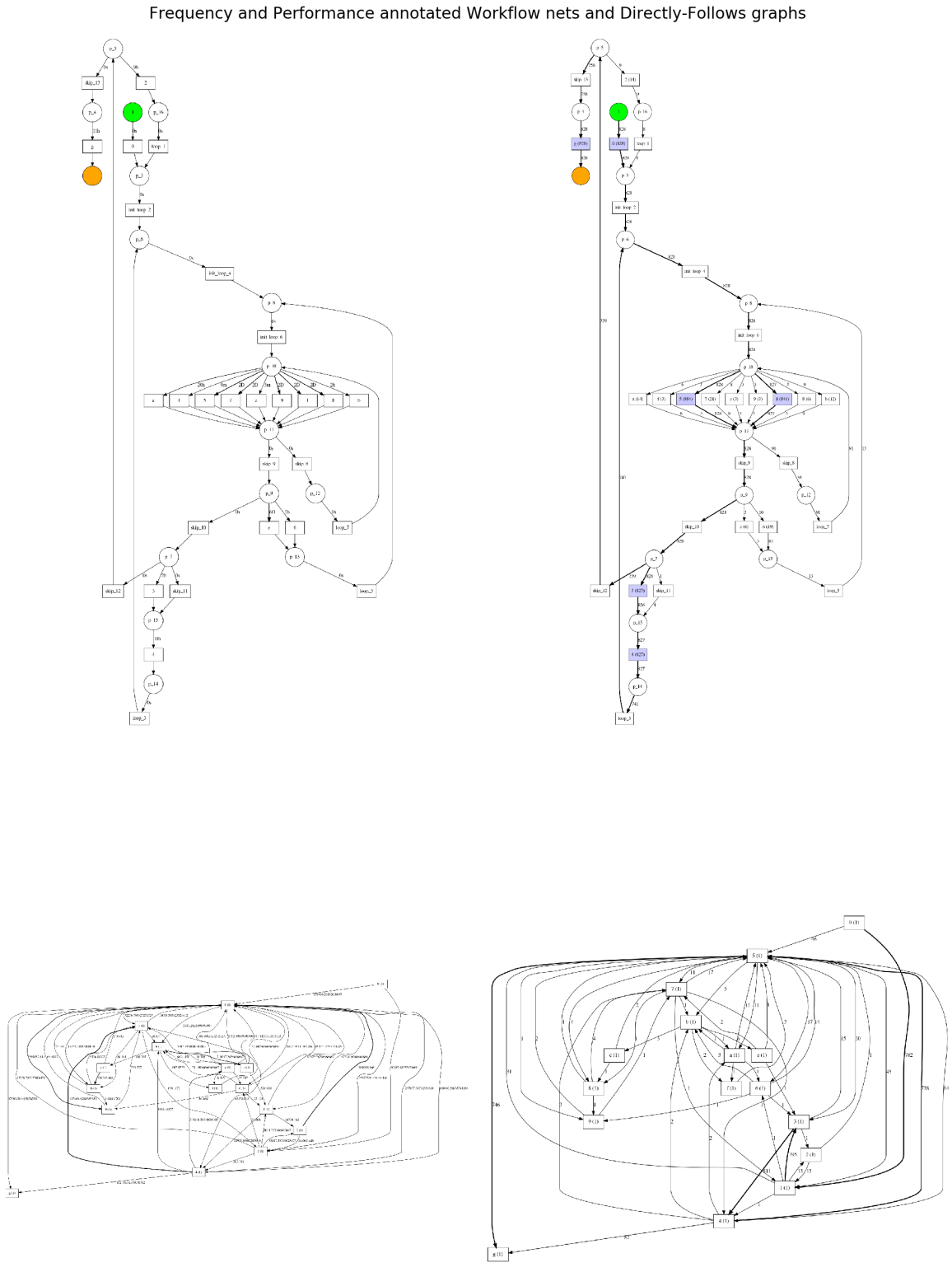


Figure 7: The petri nets (top) and directly follows graphs (bottom) annotated with average times (left) and transition frequency (right)

Directly Follows graph and Workflow Net Not all behaviour in the log is captured in this net. A decreasing factor as described above is used to filter the starting and ending activities of the traces. This forces the traces to begin and end like $[0, \ldots, g]$. This cleans up a significant amount of behaviour in the log which can be interpreted as traces that fail to start or complete properly, eliminating a source of noise. Some transitions come from low-frequency activities but are allowed as they begin and end with the allowed beginnings and endings. In addition, some very common variants (example, [0], a trace that is started but nothing else, the third most common variant), are removed, which removes many of the "skip" invisible transitions. Allowing for these activities creates more generality in the model, but removes the ability to account for all but "clean" processes. Less of the behaviour in the log is skippable. The large looped `XOR` split containing the events 1 and 5, still exists. Significant numbers of visible transitions have a high frequency in the replay and as such indicate that "rules" in the data have been found, and where invisible transitions occur, they are used to connect important visible transitions.

	Values
Fitness	0.894928
Simplicity	0.644444
Generalisation	0.796445
Ratio of traces used	0.577406
Percentage of most frequent variants	1.000000
Decreasing Factor	0.500000

The model fits on most of the data using 57% of the data, of which 89% of traces fit the data. The simplicity and generality of the model are less good than the model above, considering the aim of fitting to less data was to simplify the model at the cost of generality.

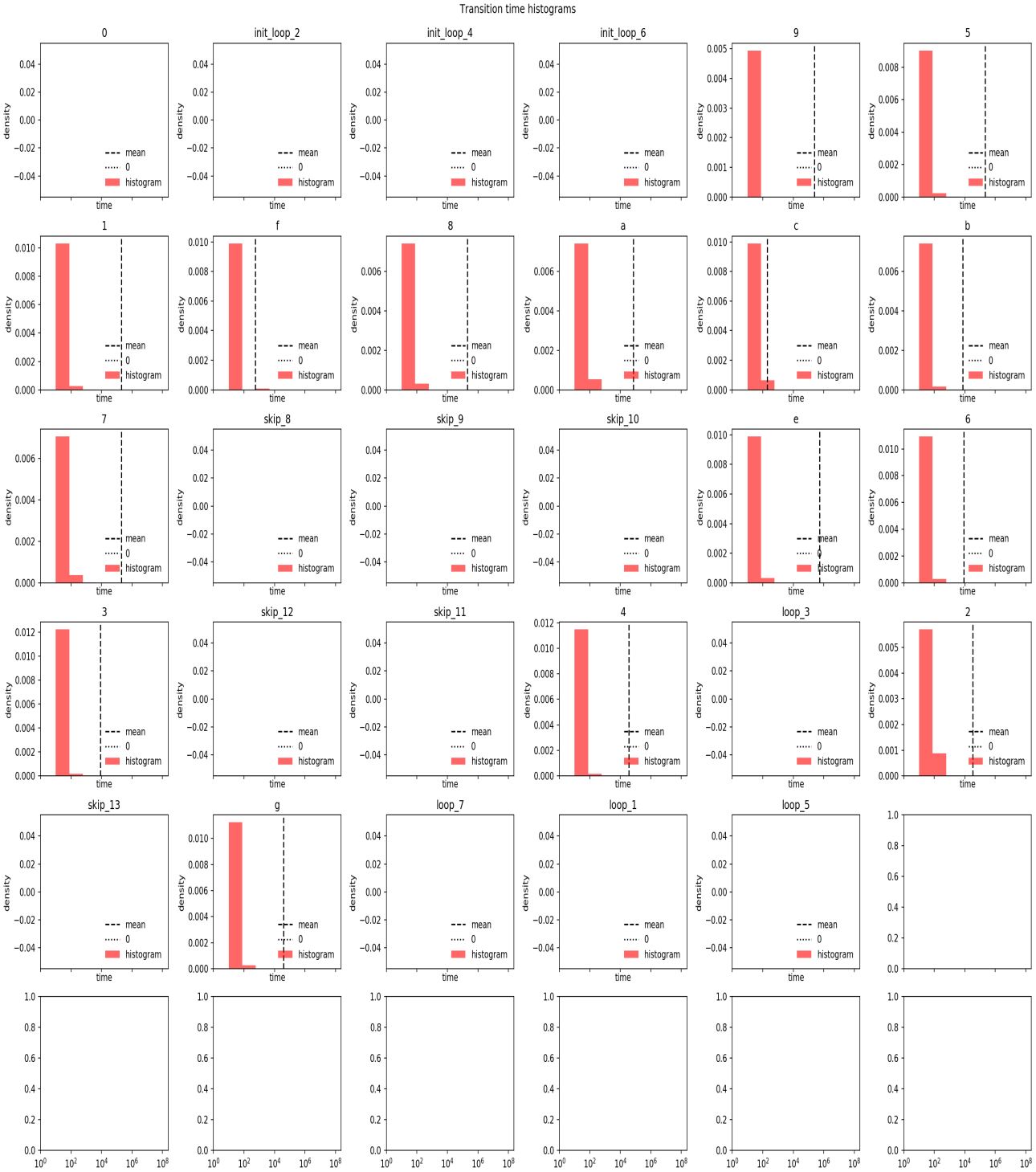


Figure 8: The histograms of the times taken to cross a particular transition.

	Transition Frequency	Minimum time	Maximum time	Mean time	Transition rate lower bound	Transition rate upper bound	$R\hat{2}$	$P(\lambda) = 0$	Passes Anderson-Darling test at 15pc significance
9	0.001693	36.169	5.988228e+05	249223.573000	-6.213507e-07	0.000009	0.911163	3.004173e-01	True
5	0.494684	9.145	2.323918e+07	215484.246826	4.321498e-06	0.000005	0.851338	2.080886e-08	False
1	0.474605	12.471	9.588298e+06	204042.189831	3.628474e-06	0.000006	0.842108	2.163355e-01	False
f	0.001693	16.415	1.593023e+03	544.087333	-2.846147e-04	0.003964	0.843646	4.085207e-01	True
8	0.003386	16.796	6.303127e+05	205886.903667	8.921739e-07	0.000009	0.834680	1.747192e-01	True
a	0.007901	19.589	5.182759e+05	72895.112714	6.391965e-06	0.000021	0.845681	1.397225e-01	False
c	0.001693	56.362	4.669490e+02	198.395000	-7.805400e-04	0.010872	0.901209	2.777816e-01	True
b	0.006772	28.134	5.911710e+04	7286.397500	5.806331e-05	0.000217	0.886567	1.625266e-01	False
7	0.011851	10.263	2.172504e+06	203936.310667	2.766196e-06	0.000007	0.851704	1.213448e-01	False
e	0.003699	23.785	3.113940e+06	519047.262667	3.538925e-07	0.000004	0.822243	3.631594e-01	True
6	0.011714	17.359	9.074481e+04	8662.102105	6.253790e-05	0.000169	0.832362	1.624465e-01	False
3	0.517198	9.010	1.631252e+06	8307.421139	1.121045e-04	0.000129	0.844548	6.482975e-04	False
4	0.897937	8.647	1.223770e+06	36276.907567	2.567428e-05	0.000030	0.842612	7.259987e-12	False
2	0.018135	14.017	4.227125e+05	33117.587769	1.345947e-05	0.000047	0.840306	3.278715e-01	False
g	1.000000	11.670	3.618586e+06	40293.400741	2.303812e-05	0.000027	0.837518	1.675189e-05	False

Activities The histograms of the activity times that a very sharp exponential function may be fittable to this data. The tails of the fit distribution hold the most disagreement with the data. In most cases, the data is skewed right and where it isn't, it is usually unskewed.

The CDF plots show that that while the fits visually agree quite well with the data $R^2 > 0.8$, the tails of the distributions mostly disagree with the Anderson-Darling null hypothesis (at a 15% significance level) that the data was sampled from an exponential distribution. It is worth noting that most of the cases where the null hypothesis was not rejected, the sample size was small, the sample error of the rate parameter λ_i was large, which allowed for some "plausible acceptability" that the samples came from an exponential distribution.

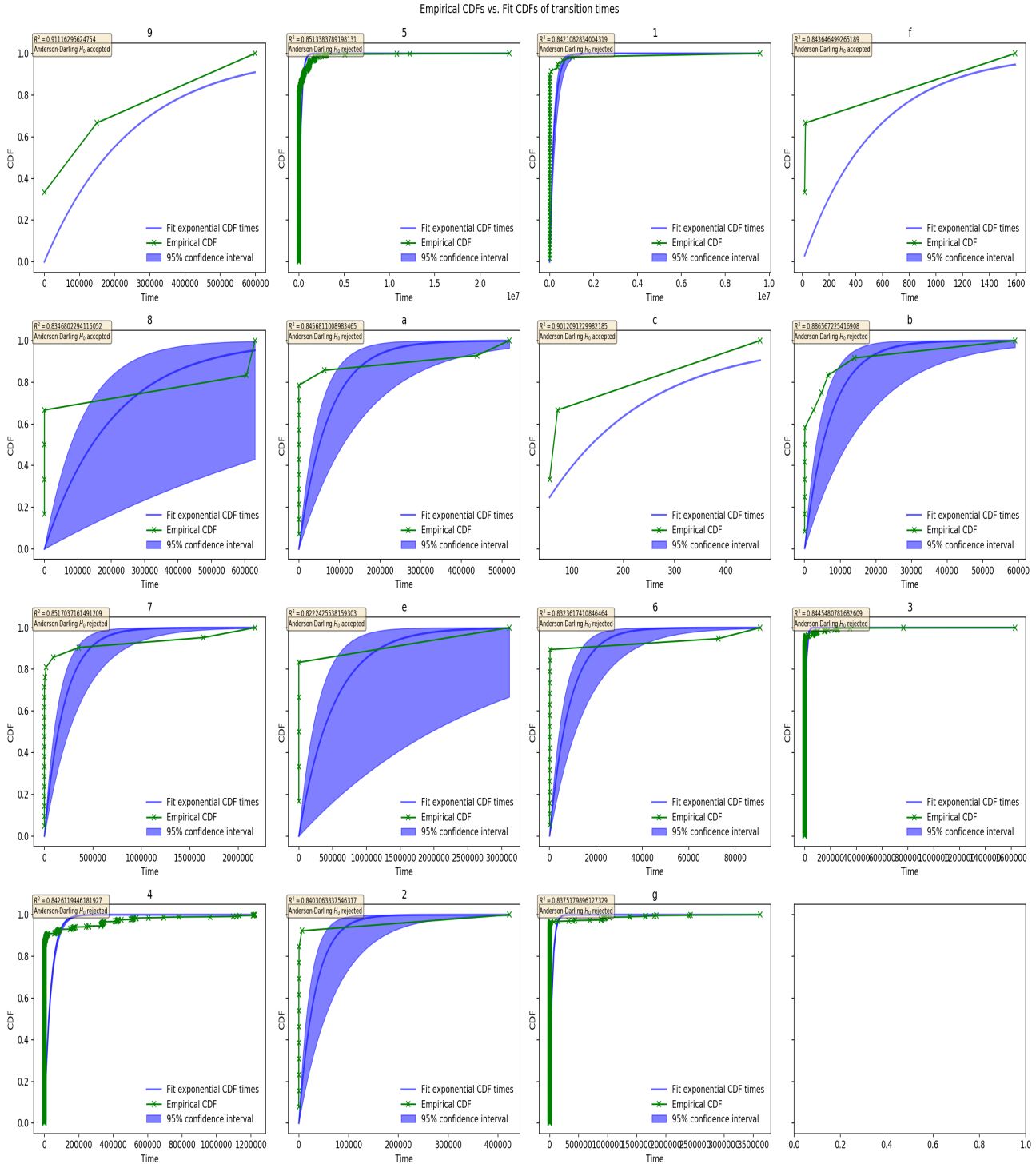


Figure 9: The empirical CDFs of the times taken to cross a particular transition.

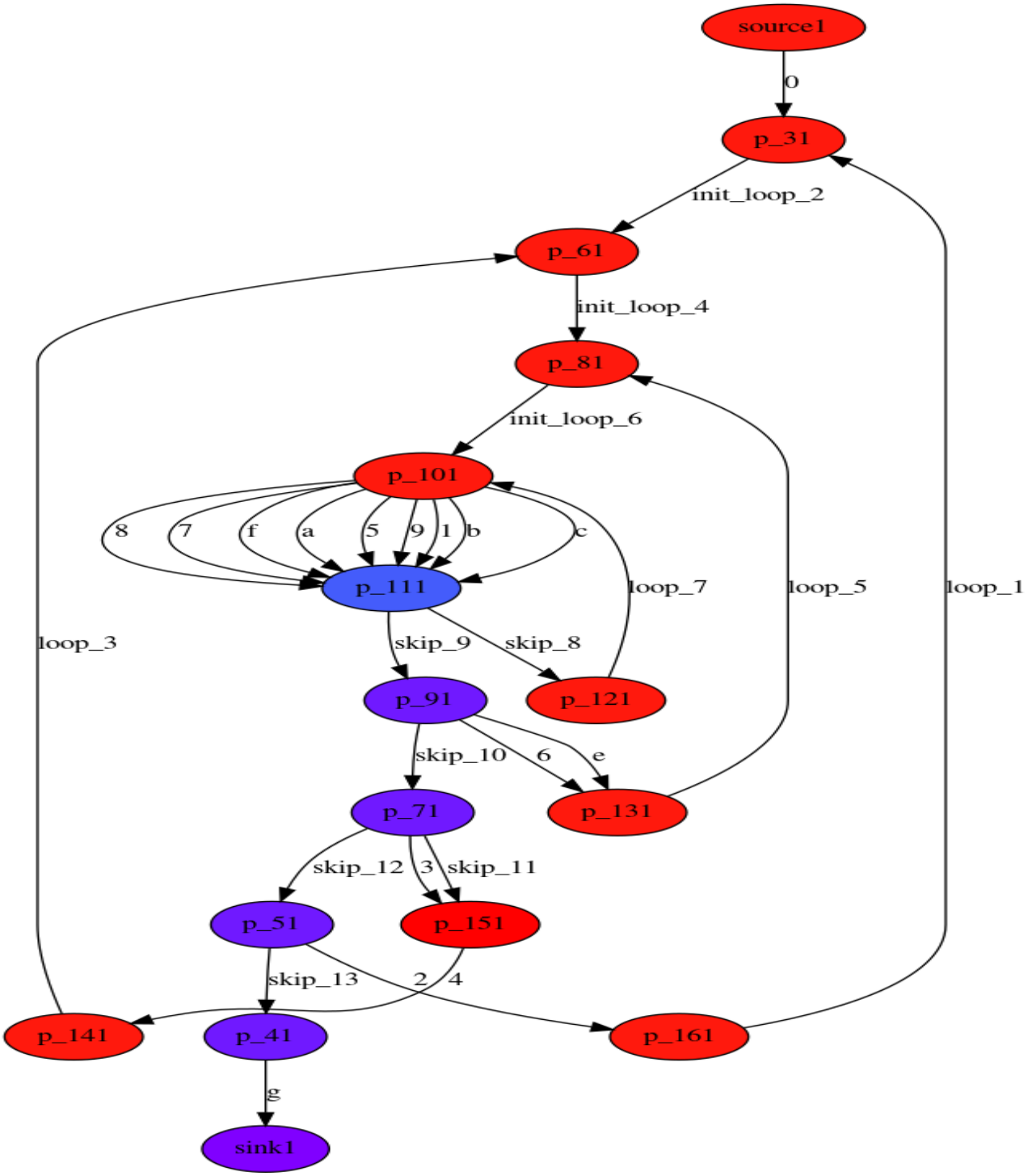


Figure 10: A reachability graph coloured by the mean hitting time it would take to get from this state to the end state `sink1`. Denoted $E_i[T_A] = k_i, A$ in the theory section, the more blue areas are close to 0 and the more red areas are close to the maximum of $E_i[T_A]$.

Modelled time-evolution The above diagram shows the transition system of the workflow net above. Two areas of roughly equal finish time can be identified. The red area indicates the states of the workflow net after the transition 0 which contain most of the longer transitions - roughly divided by the large `OR` split and the states where the `OR` split can be re-entered. The set of transitions between `p_101` and `p_111` fires quite slowly but still contains transitions $5, 1$ which are quite frequent. One notable difference between this and the model without the decreasing factor is that the transition j is no longer present. This rare activity takes up quite a lot of time and without it, the process is able to complete more quickly. The transitions $\{0, g\}$ correspond lead to start and end states cleanly. Most invisible transitions have a high transition rate to allow them to not affect the overall time evolution.

Figure 11: The reachability graph of the workflow net above. the more blue areas are close to 0 and the more red areas are close to the maximum of $E_i[T_A]$.

variant	Count	Fitness	Path Probability	Sum of mean act. times	se(Mean act. times)	Mean Case Duration	P(Mean act. times = Mean Case Duration)
0,1,3,4,5,g	713	1	1.06241e-08	503900	1.01302e+07	377880	0.00657665
0,5,1,3,4,g	40	0.857143	1.08075e-09	503900	1.01302e+07	475511	0.929719
0,1,5,3,4,g	28	0.857143	1.08075e-09	503900	1.01302e+07	256660	0.00768059
0,1,3,5,4,g	12	0.857143	1.06241e-08	503900	1.01302e+07	353206	0.555548
0,1,2,1,3,4,5,g	6	1	5.37103e-19	740823	1.18152e+07	771063	0.866693

Variant analysis Above is a graph of the 9 most common variants accounting for a large amount of behaviour in the log. The null hypothesis that the mean variant time is equal to the sum of the mean activity times is rejected at the 95% confidence level for most of these, but accepted for the variants $[0, 5, 1, 3, 4, g]$, $[0, 1, 3, 5, 4, g]$, $[0, 1, 2, 1, 3, 4, 5]$ and $[0, 1, 3, 4, 5, 6, 5, g]$. This null hypothesis is associated with the notion of memorylessness in the workflow net, and rejecting it implies that something in the variant implies that it's mean time is significantly different from the sum of the times that comprise it.

Fitting on 90% of variants, Decreasing Factor = 1

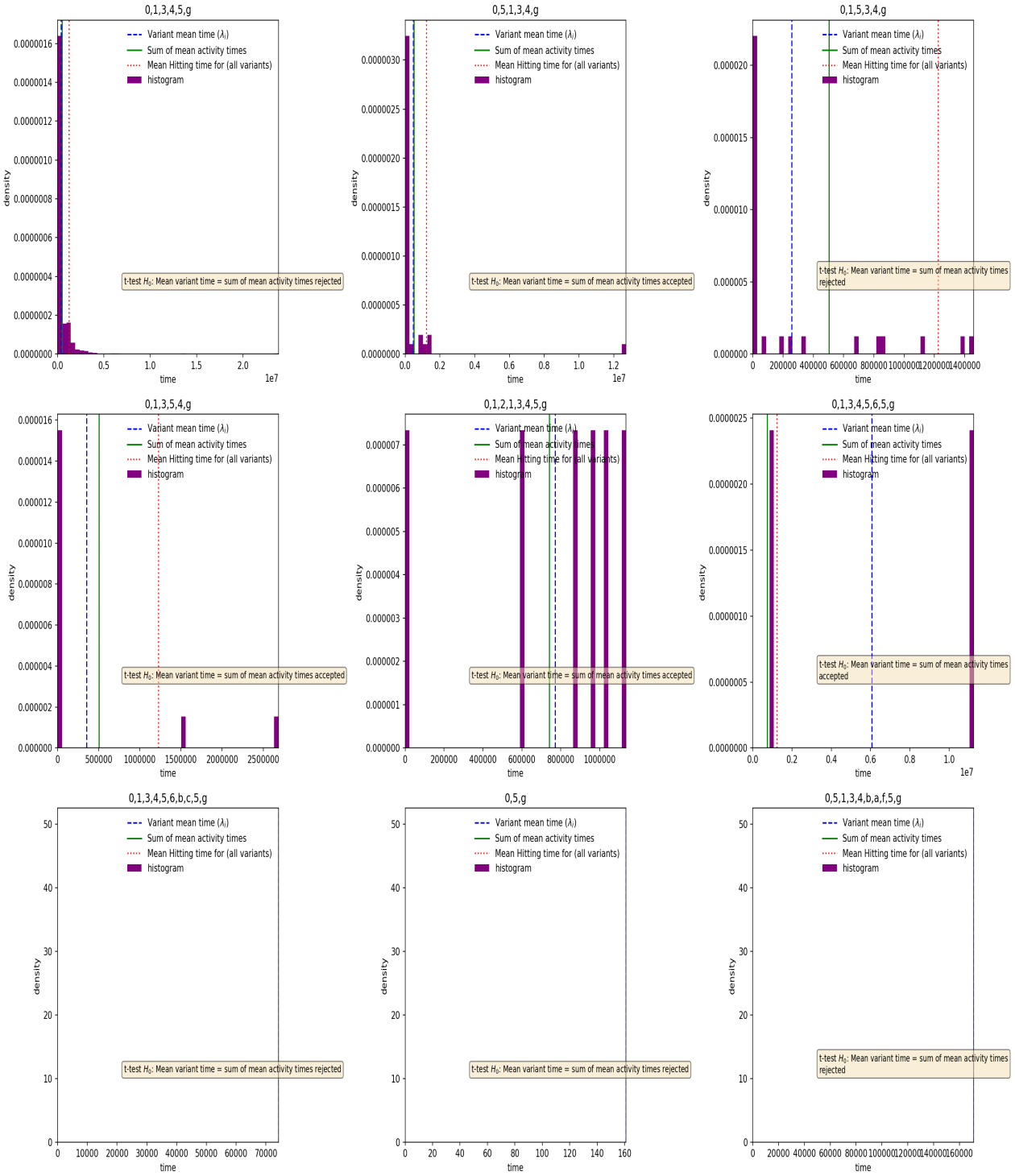


Figure 12: The histograms of time taken of the 9 most frequent variants. Labelled are the Expected hitting time $E_{start}[T_{end}]$ the sum of activity times $E_{start}[S_\sigma]$, and the sample mean of the variants.

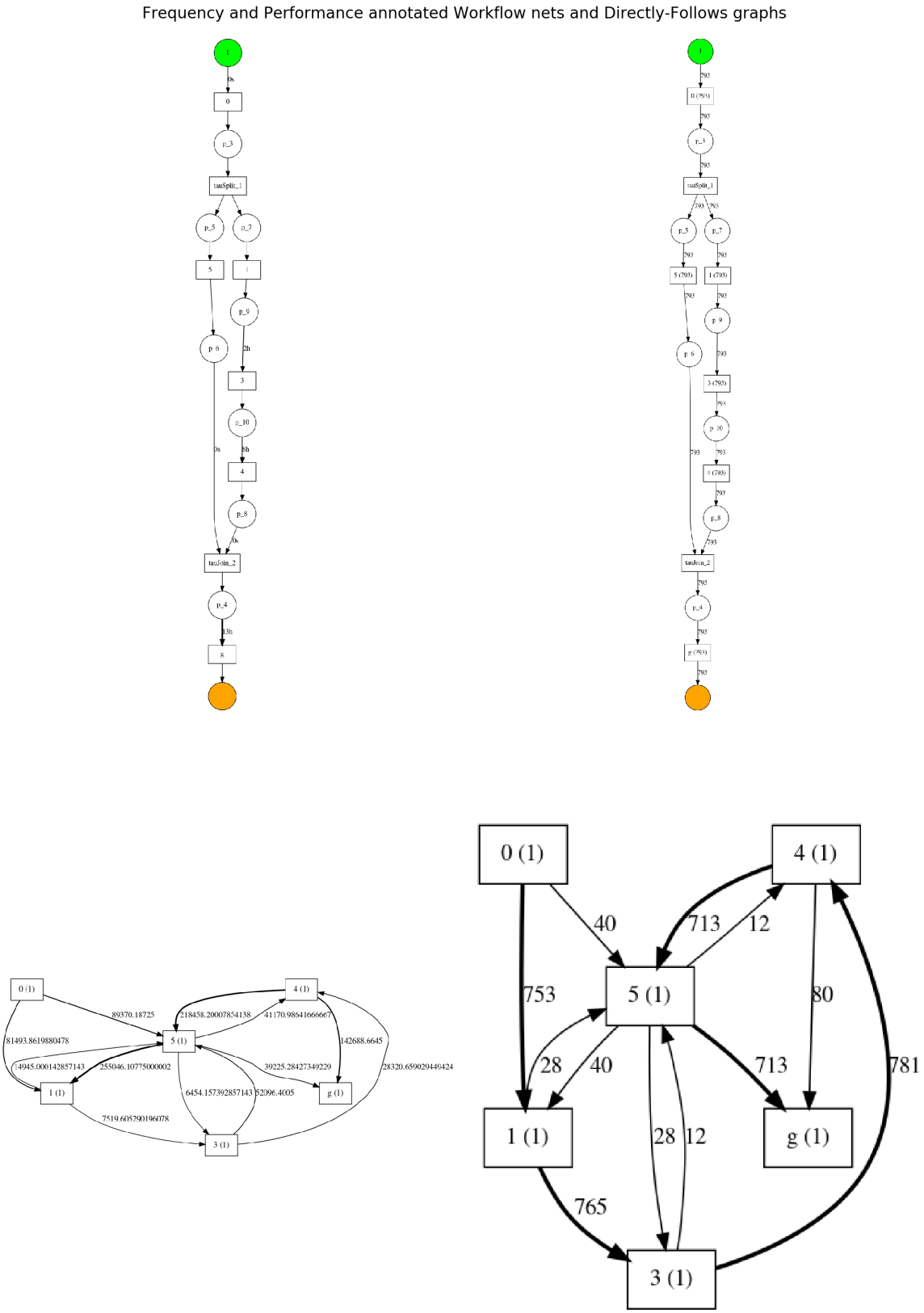


Figure 13: The petri nets (top) and directly follows graphs (bottom) annotated with average times (left) and transition frequency (right)

Directly Follows graph and Workflow Net Not all behaviour in the log is captured in this net. A decreasing factor as described above is used to filter the starting and ending activities of the traces. This forces the traces to begin and end like $[0, \ldots, g]$. This cleans up a very large amount of behaviour in the log, only some of which can be interpreted as traces that fail to start or complete properly. The `XOR` split containing the events 1 and 5, still exists but in a smaller form, now represented in a way that clearly shows concurrency between 5 and $\{1, 3, 4\}$. Significant numbers of visible transitions have a high frequency in the replay and as such indicate that "rules" in the data have been found, and where invisible transitions occur, they are used to connect important visible transitions.

	Values
Fitness	1.000000
Simplicity	0.444444
Generalisation	0.964489
Ratio of traces used	0.552999
Percentage of most frequent variants	0.900000
Decreasing Factor	1.000000

The model fits on most of the data using 52% of the data, of which 100% of traces fit the data. However, the Simplicity metric here punishes the model strongly for the number of places relative to the number of transitions which increases as a result of the `XOR` split. The generality for the model with respect to the filtered data is high.

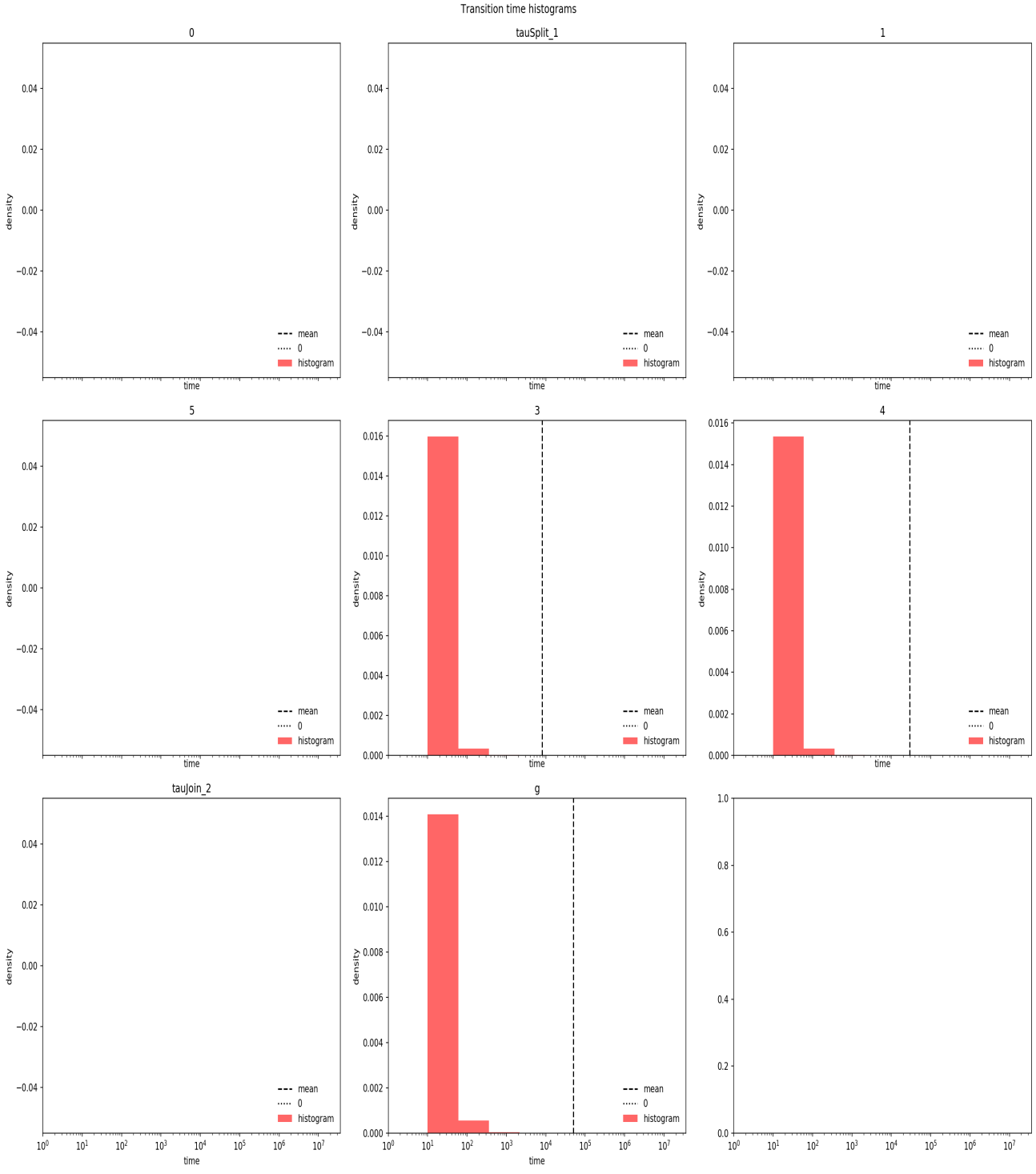


Figure 14: The histograms of the times taken to cross a particular transition.

	Transition Frequency	Minimum time	Maximum time	Mean time	Transition rate lower bound	Transition rate upper bound	$R\hat{2}$	$P(\lambda) = 0$	Passes Anderson-Darling test at 15pc significance
3	0.965895	9.010	1631252.108	8009.677753	0.000116	0.000134	0.843868	1.426221e-03	False
4	0.985093	8.647	1223770.175	29303.459451	0.000032	0.000037	0.842309	1.794172e-09	False
g	1.000000	11.670	3618586.270	49662.951888	0.000019	0.000022	0.837641	4.809341e-07	False

Activities The histograms of the activity times show that a very sharp exponential function may be fittable to this data. The tails of the fit distribution hold the most disagreement with the data.

The CDF plots show that that while the fits visually agree quite well with the data $R^2 > 0.83$, the tails of the distributions disagree with the Anderson-Darling null hypothesis (at a 15% significance level) that the data was sampled from an exponential distribution. This show that there was not enough activity in the tail ends of the distribution to justify an exponential distribution for these activities.

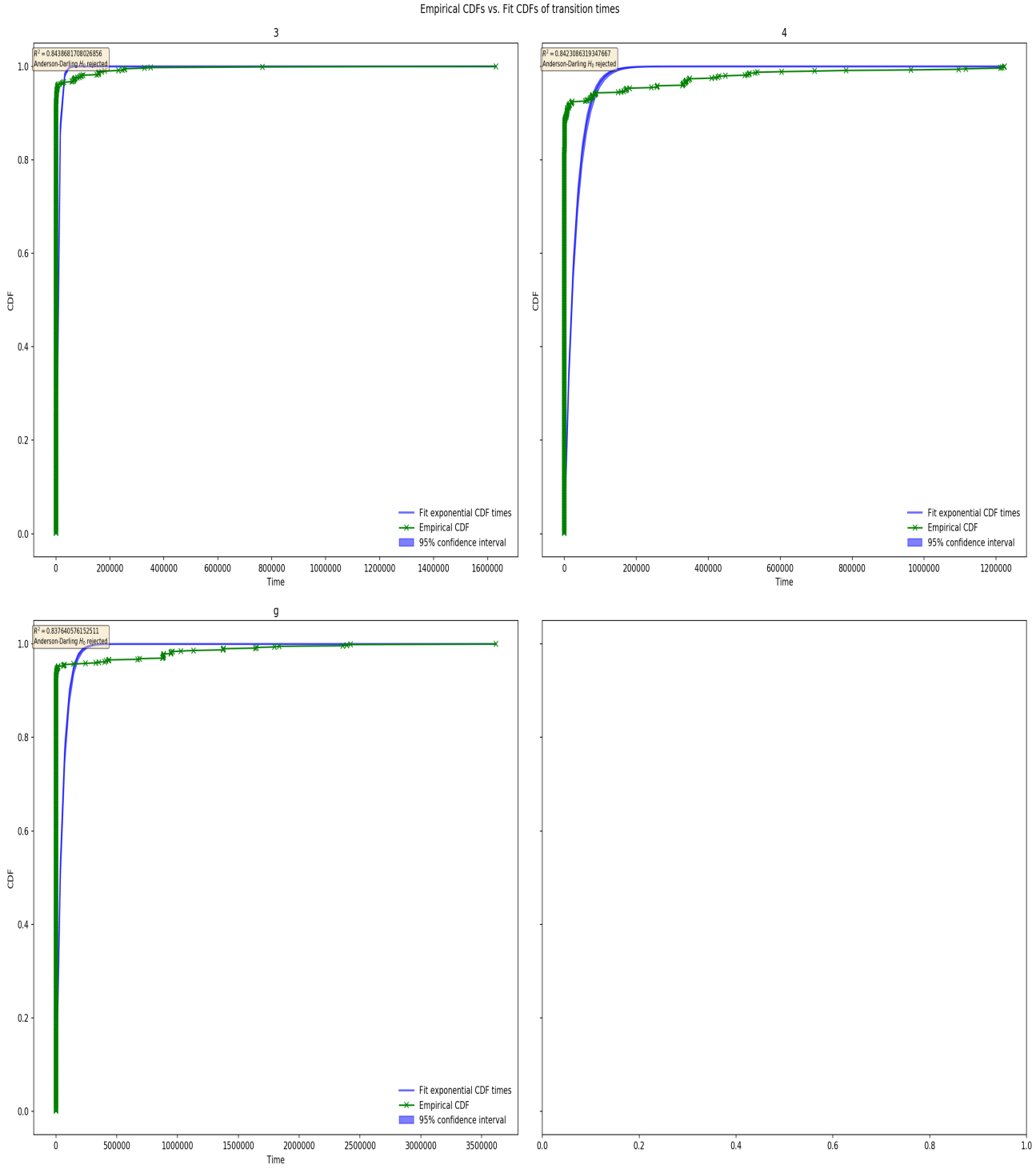


Figure 15: The empirical CDFs of the times taken to cross a particular transition.

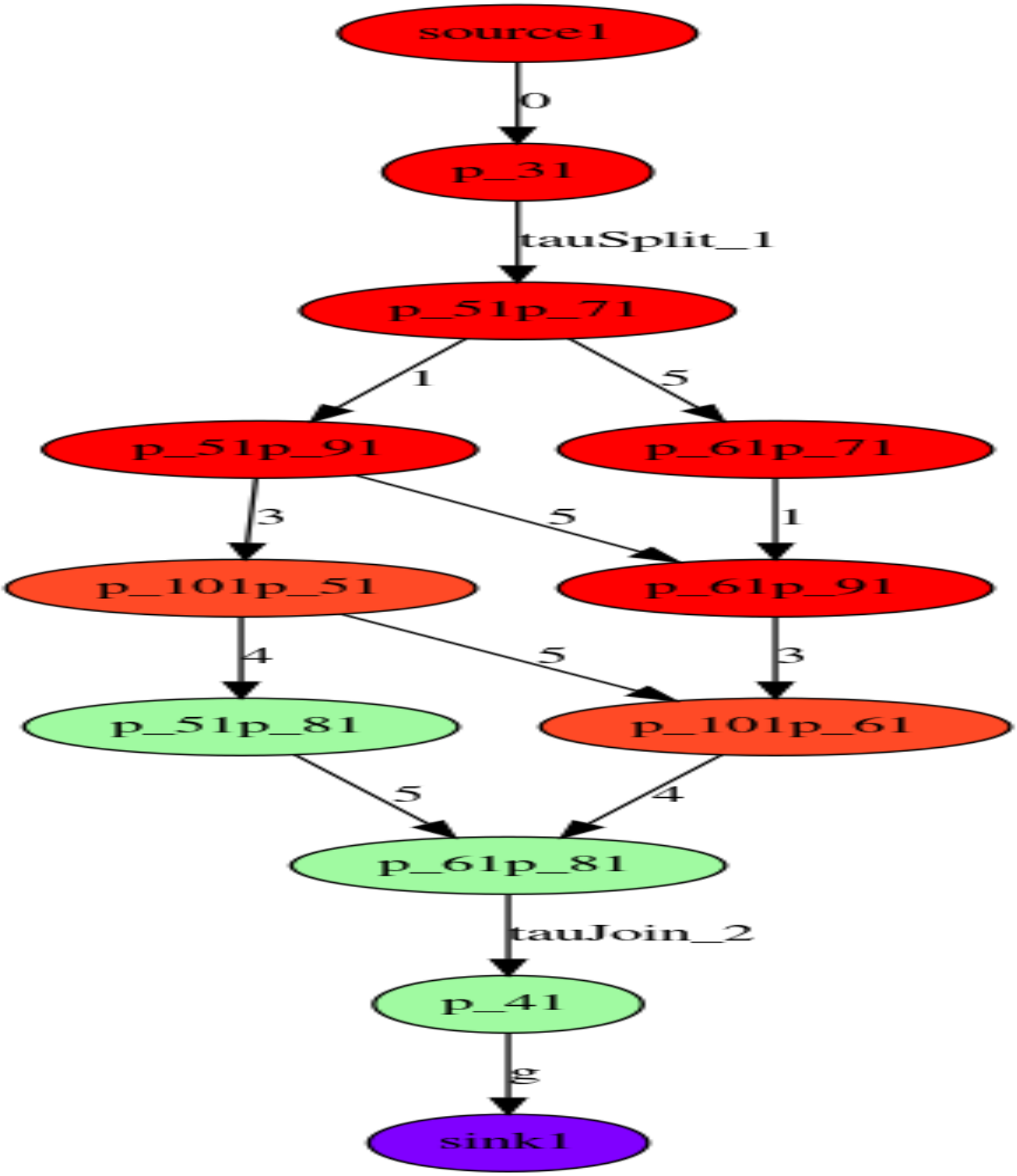


Figure 16: A reachability graph coloured by the mean hitting time it would take to get from this state to the end state `sink1`. Denoted $E_i[T_A] = k_i, A$ in the theory section, the more blue areas are close to 0 and the more red areas are close to the maximum of $E_i[T_A]$.

Modelled time-evolution The above diagram shows the transition system of the workflow net above. Two areas of roughly equal finish time can be identified. The red area indicates the states of the workflow net after the transition 0 which contain most of the longer transitions - roughly divided by "transitions before 4" in red, and "transitions after 4" in green, showing that 4 is a bottleneck in this process. This rare activity takes up quite a lot of time and without it, the process is able to complete more quickly. The transitions $\{0, g\}$ correspond lead to start and end states cleanly. The only invisible transitions here are those involved in setting up the `OR` split.

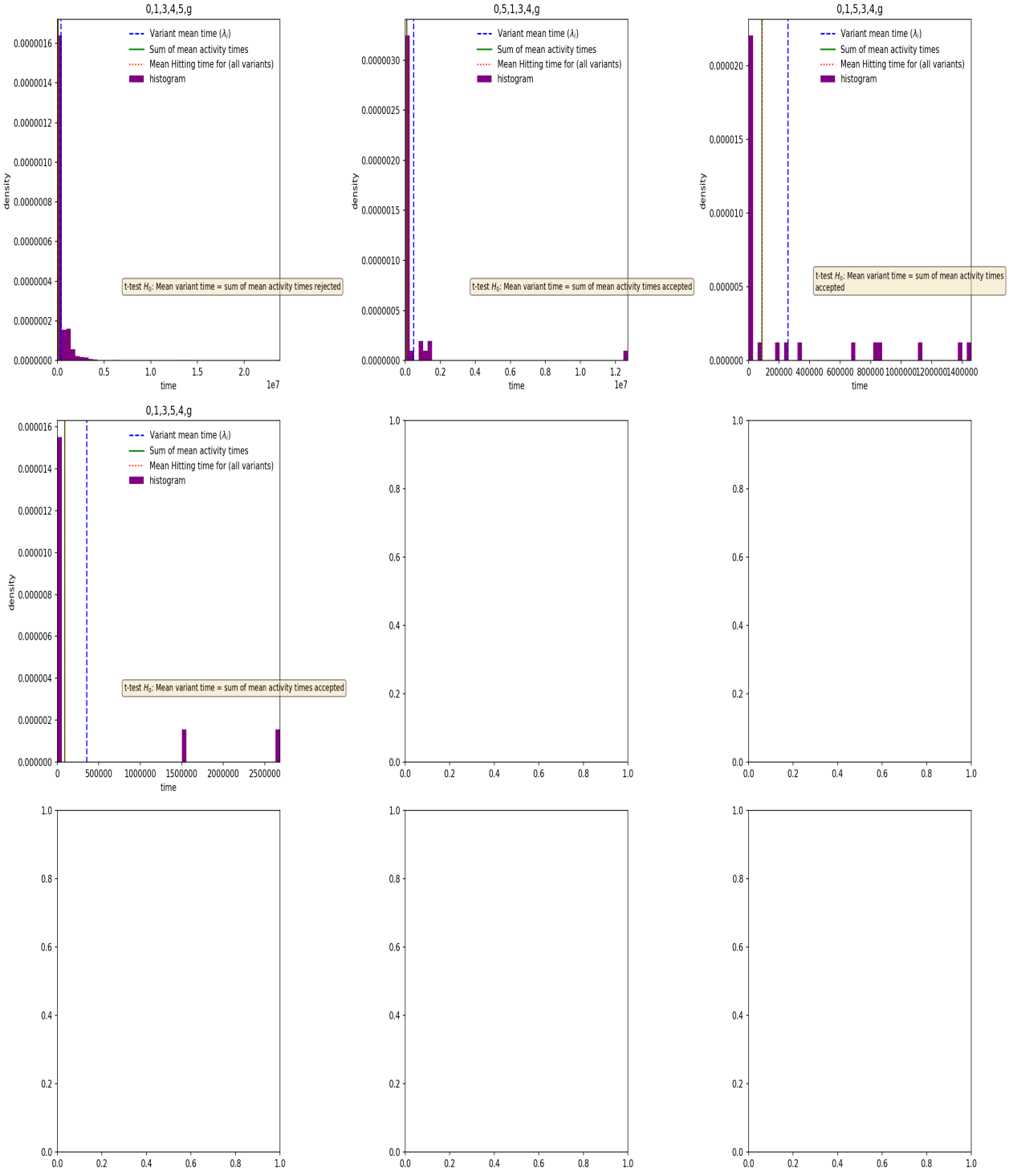


Figure 17: Histograms of the 9 most frequent variants of traces in the data. Various landmark times are shown above.

variant	Count	Fitness	Path Probability	Sum of mean act. times	se(Mean act. times)	Mean Case Duration	P(Mean act. times = Mean Case Duration)
0,1,3,4,5,g	713	1	0.218234	86889.2	2.44682e+06	377880	5.42046e-10
0,5,1,3,4,g	40	1	0.218234	86889.2	2.44682e+06	475511	0.231614
0,1,5,3,4,g	28	1	0.218234	86889.2	2.44682e+06	256660	0.058206
0,1,3,5,4,g	12	1	0.218234	86889.2	2.44682e+06	353206	0.30562

Variant analysis Above is a graph of the 9 most common variants accounting for a large amount of behaviour in the log. The null hypothesis that the mean variant time is equal to the sum of the mean activity times is accepted for accepted for the variants $[0, 5, 1, 3, 4, g]$, $[0, 1, 3, 5, 4, g]$, and $[0, 1, 5, 3, 4, g]$. This null hypothesis is associated with the notion of memorylessness in the workflow net, and rejecting it implies that something in the variant implies that it's mean time is significantly different from the sum of the times that comprise it.

5.2 Metadata from an Energy trading company

This event log is comprised of metadata on the data collection logs of an energy trading company. The structure is similar to the dataset above, but is much larger (159Mb vs. 2Mb)

Fitting on 50% of the traces, Decreasing Factor = 0

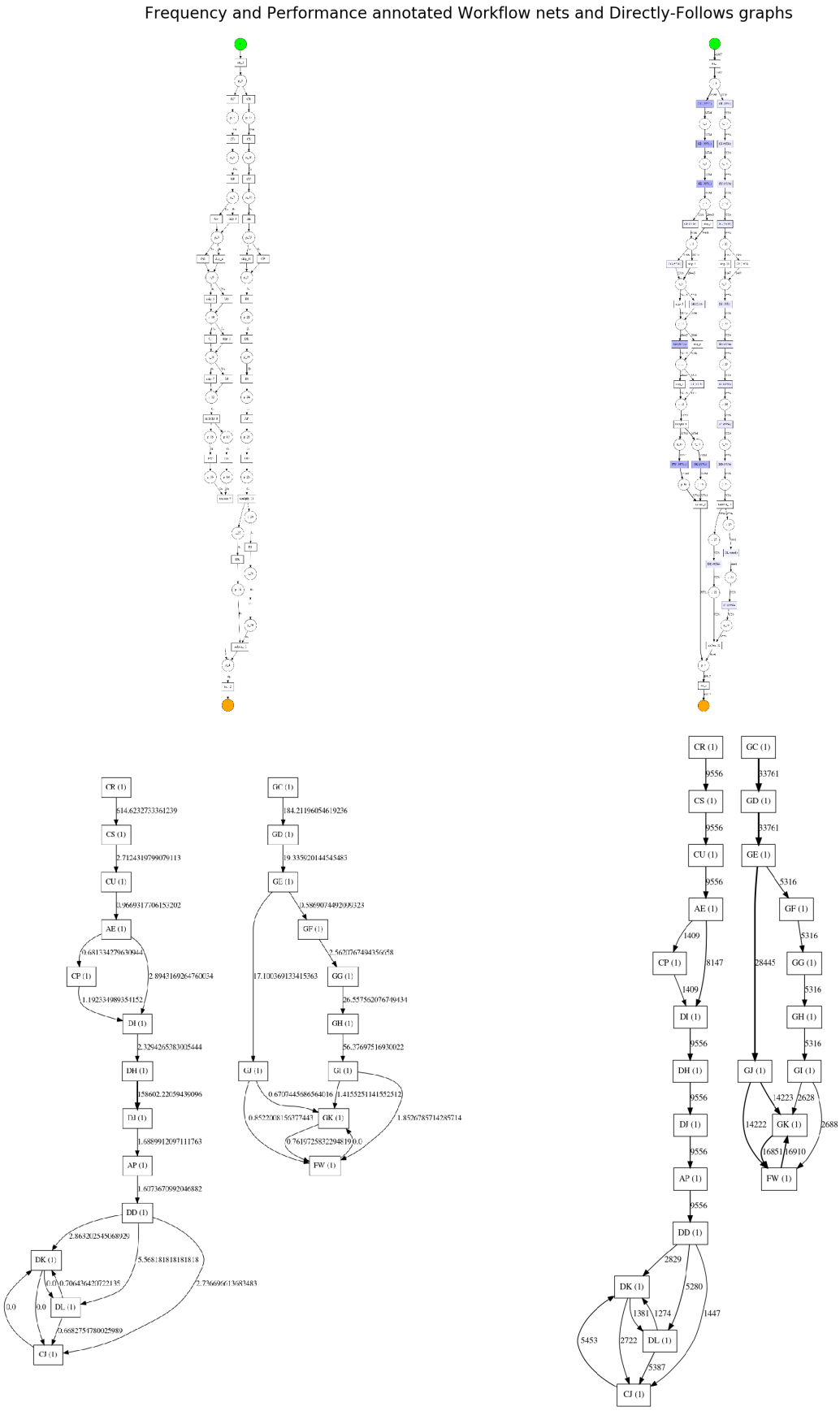


Figure 18: The petri nets (top) and directly follows graphs (bottom) annotated with average times (left) and transition frequency (right)

Directly Follows graph and Workflow Net Not all behaviour in the log is captured in this net. The top 50% most frequent variants in this dataset were used to construct a process model. From these diagrams, it is clear that two different processes can be discovered:

- $[GC, \ldots, FW]$ and $[GC, \ldots, GK]$
- $[CR, \ldots, CJ]$ and $[CR, \ldots, DK]$

Additionally, the directly-follows graphs make it clear that two different chains of events are occuring. This would suggest more success may be had in confining future mining projects to one branch of this chain.

	Values
Fitness	0.933167
Simplicity	0.530303
Generalisation	0.990603
Ratio of traces used	0.513648
Percentage of most frequent variants	0.500000
Decreasing Factor	0.000000

The model fits on most of the data using 51% of the data, of which 93% of traces fit the data. The model has relatively simple behaviour and clearly describes the majority of the data in the filtered log. However, the two different processes are joined unnecessarily, creating more places and complexity than required, resulting in the simplicity model being lower than expected.

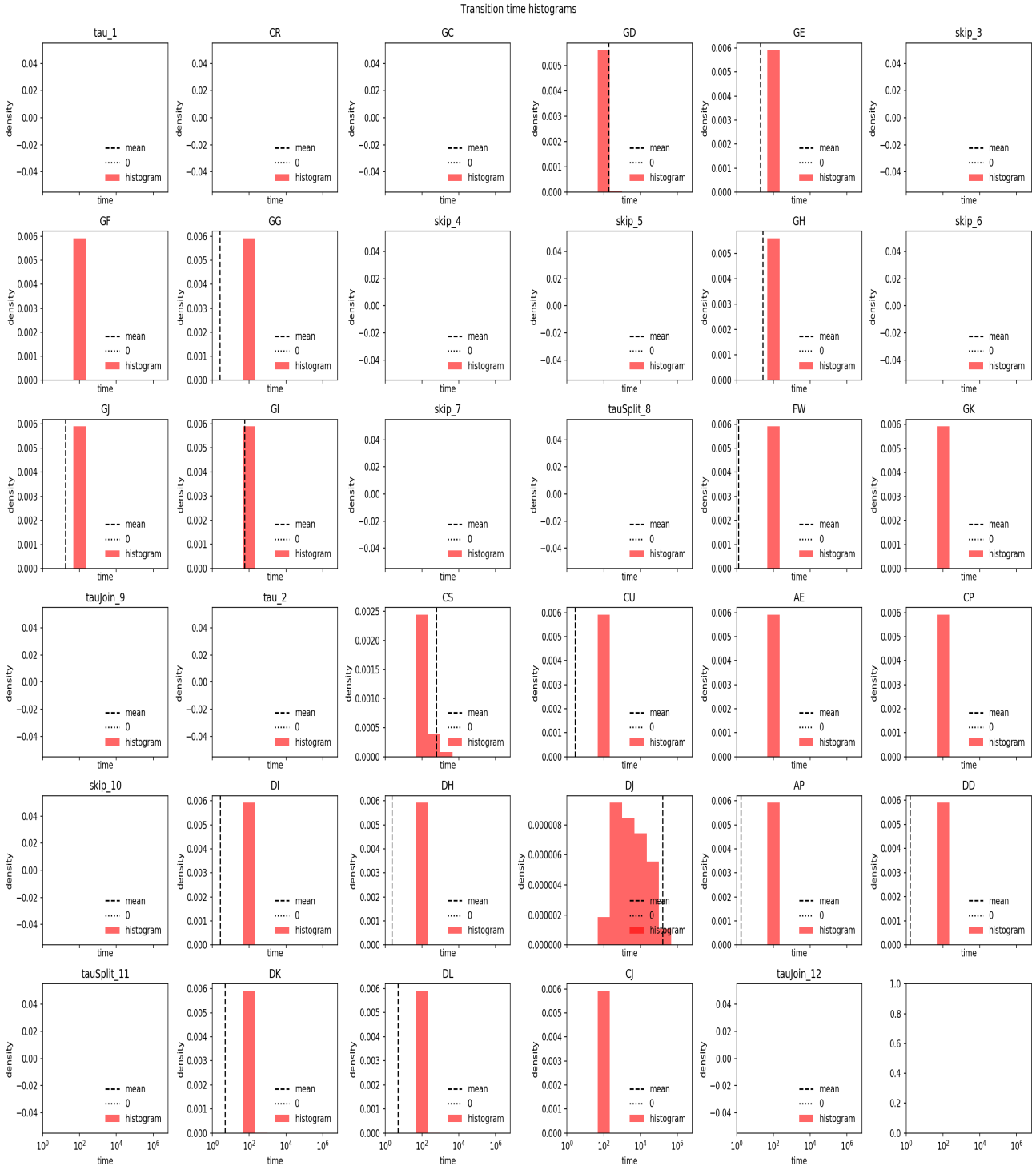


Figure 19: The histograms of the times taken to cross a particular transition.

	Transition Frequency	Minimum time	Maximum time	Mean time	Transition rate lower bound	Transition rate upper bound	$R\hat{2}$	$P(\lambda = 0$	Passes Anderson-Darling test at 15pc significance
GD	1.000000	0.0	97740.0	184.211961	0.005375	0.005493	0.891038	2.744652e-29	False
GE	1.000000	0.0	240.0	19.335920	0.051205	0.052332	0.889773	0.000000e+00	False
GF	0.157460	0.0	60.0	0.586907	1.658766	1.752334	0.990500	4.908400e-13	False
GG	0.157460	0.0	120.0	2.562077	0.379982	0.401416	0.962718	1.781065e-51	False
GH	0.157460	0.0	70440.0	26.557562	0.036658	0.038726	0.994316	1.510666e-01	False
GJ	0.842540	0.0	106740.0	17.100369	0.057843	0.059231	0.894613	5.302666e-06	False
GI	0.157460	0.0	420.0	56.376975	0.017268	0.018243	0.919148	0.000000e+00	False
FW	0.667055	0.0	12840.0	1.279583	0.773771	0.790801	0.985698	8.197522e-04	False
GK	0.666279	0.0	180.0	0.899262	1.101018	1.125251	0.985726	1.424572e-111	False
CS	1.000000	0.0	16500.0	614.623273	0.001595	0.001662	0.927530	0.000000e+00	False
CU	1.000000	0.0	60.0	2.712432	0.361491	0.376592	0.960559	3.973288e-98	False
AE	1.000000	0.0	60.0	0.966932	1.014053	1.056414	0.984645	1.247168e-35	False
CP	0.147447	0.0	60.0	0.681334	1.390896	1.547456	0.989018	6.090134e-05	False
DI	1.000000	0.0	60.0	2.643365	0.370936	0.386432	0.961428	1.272540e-95	False
DH	1.000000	0.0	120.0	2.329427	0.420928	0.438511	0.965540	8.091926e-84	False
DJ	1.000000	60.0	641040.0	158602.220594	0.000006	0.000006	0.902789	0.000000e+00	False
AP	1.000000	0.0	60.0	1.688991	0.580536	0.604787	0.974137	2.761833e-61	False
DD	1.000000	0.0	240.0	1.607367	0.610016	0.635499	0.975546	8.314831e-56	False
DK	0.507165	0.0	420.0	4.809544	0.203870	0.212386	0.937955	1.400621e-158	False
DL	0.366492	0.0	300.0	5.044288	0.193579	0.203305	0.935728	9.684181e-119	False
CJ	0.882364	0.0	120.0	0.675574	1.445393	1.518012	0.989251	1.037975e-17	False

Activities The CDF plots show that that while the fits visually agree quite well with the data $R^2 > 0.88$, the distributions disagree with the Anderson-Darling null hypothesis (at a 15% significance level) that the data was sampled from an exponential distribution. The fact that all tests rejected the null hypothesis at such a low significance level does not entirely rule out the possibility of memorylessness - it should be noted that all these tests were carried out independently and the introduction of distributions where memorylessness does not hold may cause assumptions to fail further on.

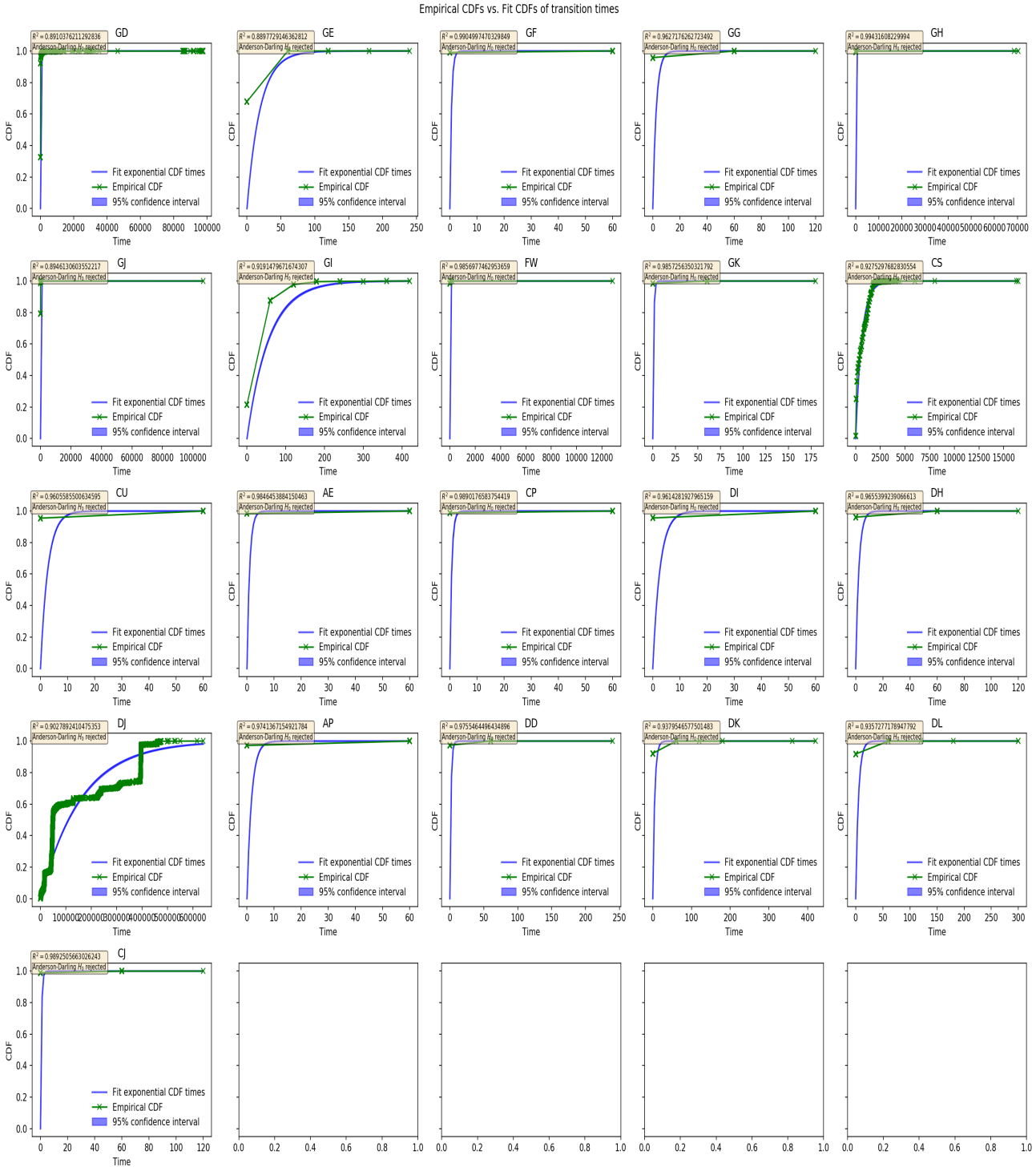


Figure 20: The empirical CDFs of the times taken to cross a particular transition.

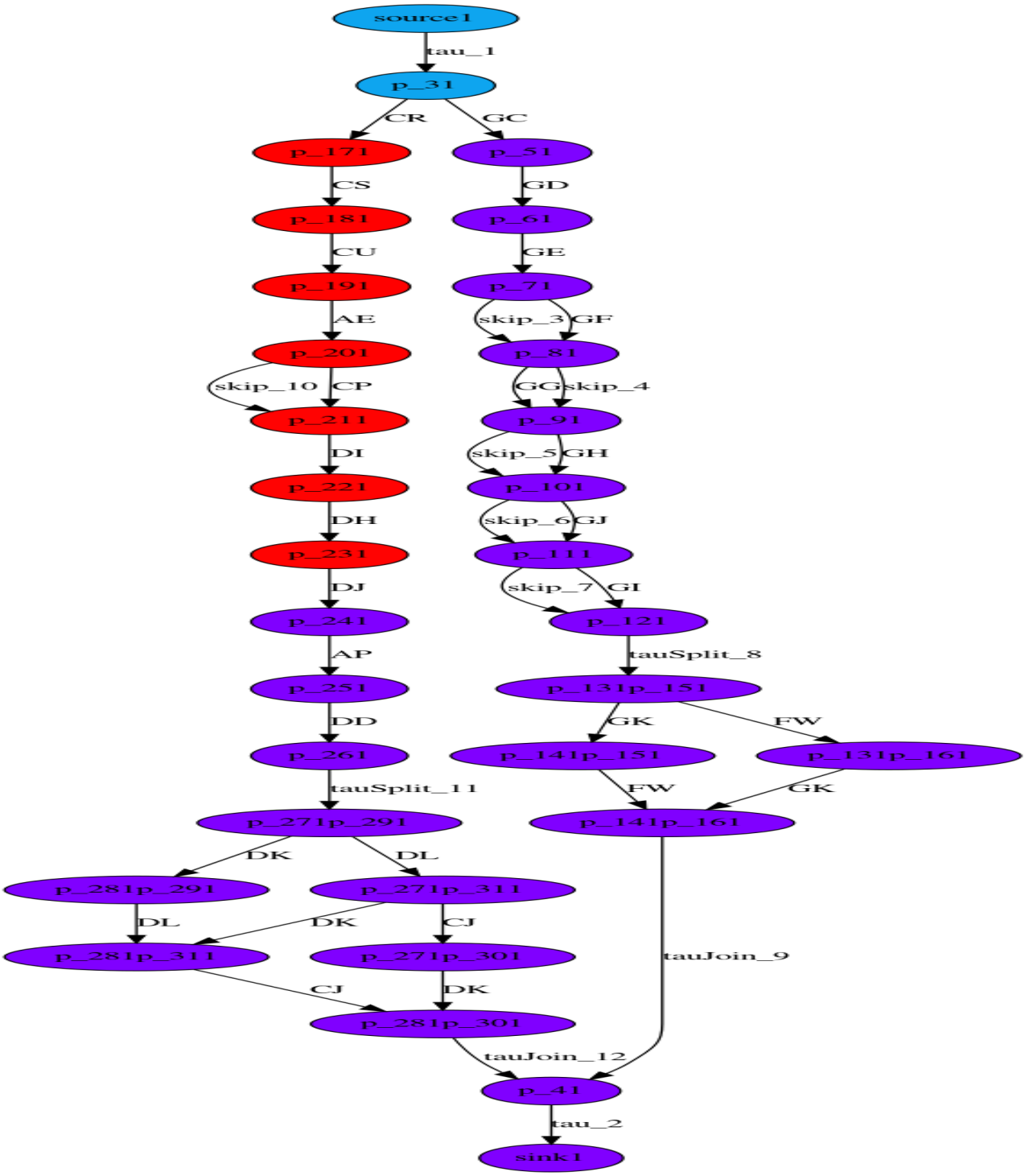


Figure 21: A reachability graph coloured by the mean hitting time it would take to get from this state to the end state `sink1`. Denoted $E_i[T_A] = k_i, A$ in the theory section, the more blue areas are close to 0 and the more red areas are close to the maximum of $E_i[T_A]$.

Modelled time-evolution The above diagram shows the transition system of the workflow net above. The $[CR, \ldots, CJ]$ and $[CR, \ldots, DK]$ process chains complete much slower than the other chains. This is mostly due to the $[CR, \ldots]$ chains having to pass through DJ, which has a mean completion time orders of magnitude higher than the rest of the activities. In addition, only 22% of traces pass through the $[CR, \ldots]$ chain, meaning by $t = 1000$ in the diagram below, most traces have completed.

variant	Count	Fitness	Path Probability	Sum of mean act. times	se(Mean act. times)	Mean Case Duration	P(Mean act. times = Mean Case Duration)
GC,GD,GE,GJ,GK,FW	14223	1	0.779394	222.605	40644.1	203.544	0.432868
GC,GD,GE,GJ,FW,GK	14222	1	0.779394	222.605	40644.1	245.113	0.438304
GC,GD,GE,GF,GG,GH,GI,FW,GK	2688	1	0.779394	291.519	44033	292.121	0.990878
GC,GD,GE,GF,GG,GH,GI,GK,FW	2628	1	0.779394	291.519	44033	263.699	0.57936
CR,CS,CU,AE,DI,DH,DJ,AP,DD,DL,CJ,DK	2597	1	0.0832699	159080	1.55508e+07	56119.8	0
CR,CS,CU,AE,DI,DH,DJ,AP,DD,DK,CJ	1448	0.916667	0.204126	159075	1.55504e+07	330501	2.70982e-294
CR,CS,CU,AE,DI,DH,DJ,AP,DD,CJ,DK	1447	0.916667	0.204126	159075	1.55504e+07	325050	5.06125e-272
CR,CS,CU,AE,CP,DI,DH,DJ,AP,DD,DL,CJ,DK	1409	1	0.0832699	159081	1.55508e+07	195501	9.82738e-20
CR,CS,CU,AE,DI,DH,DJ,AP,DD,DK,DL,CJ	1381	1	0.0832699	159080	1.55508e+07	63713.7	3.11622e-318
CR,CS,CU,AE,DI,DH,DJ,AP,DD,DL,DK,CJ	1274	1	0.0832699	159080	1.55508e+07	49867	0

Variant analysis Above is a graph of the 9 most common variants in the log. The null hypothesis that the mean variant time is equal to the sum of the mean activity times is rejected at the 95% confidence level by the traces in the $[CR, \ldots]$ chain, and accepted by the traces in the $[GC, \ldots]$ chain. This null hypothesis is associated with the notion of memorylessness in the workflow net, and accepting it implies that the process is memoryless. The claim that the $[GC, \ldots]$ chain is memoryless is further investigated later, as it contrasts with the implications of the Anderson-Darling tests above.

Fitting on 100% of the traces, Decreasing Factor = 1

Directly Follows graph and Workflow Net Only variants beginning as $[GC, \ldots]$ in this event log were used to construct a process model. This process consists of a looped `OR` structure and very little else. While some are more frequent, the directly follows graph without the nodes GC, FX is strongly connected, meaning that it is possible to get to any node from any node. While some of the relations between activities are infrequent, there is no clear visual or causal relationship between the nodes in this graph beyond the idea of "choose a number of these activities, then stop abruptly". The causal chain seen in $[GC, \ldots]$ in the previous petri net is not seen here as more data is used.

	Values
Fitness	1.000000
Simplicity	0.766667
Generalisation	0.930086
Ratio of traces used	0.476865
Percentage of most frequent variants	1.000000
Decreasing Factor	1.000000

This model completely fits the data, for what data it uses. This model generalises out to most behaviour in the log and is quite simple in terms of place numbers and complexity.

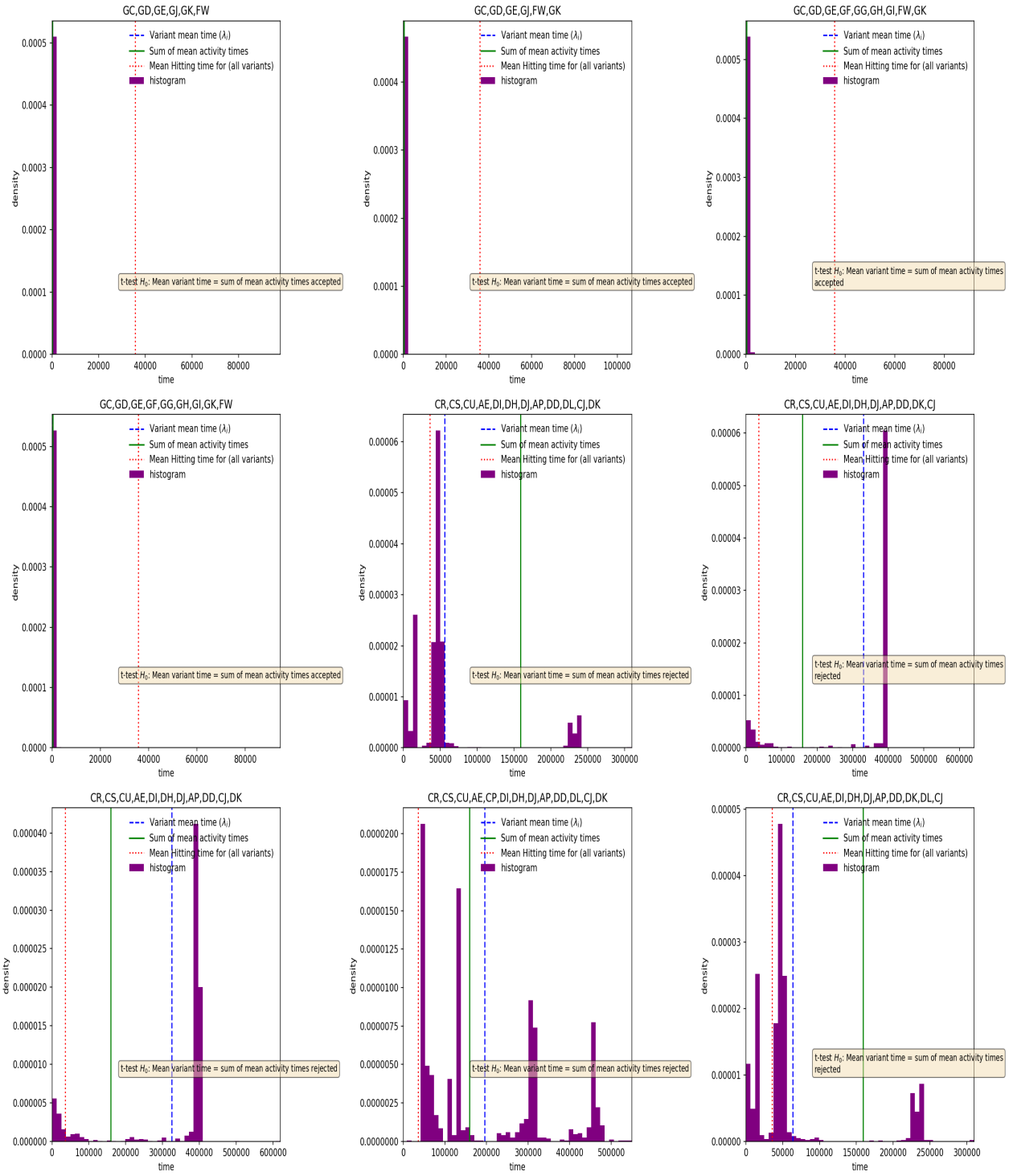


Figure 22: Histograms of the 9 most frequent variants of traces in the data. Various landmark times are shown above.

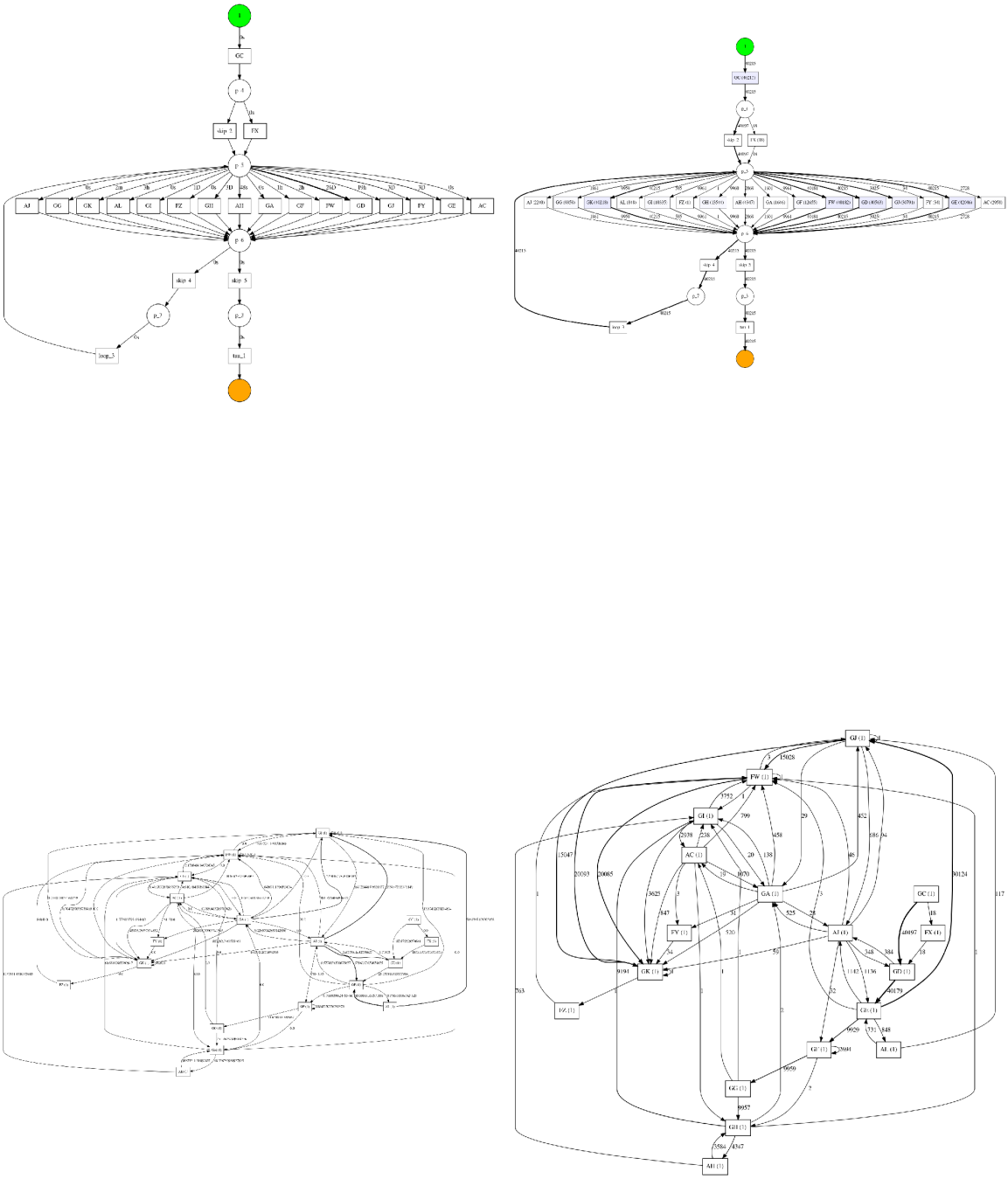


Figure 23: The petri nets (top) and directly follows graphs (bottom) annotated with average times (left) and transition frequency (right)

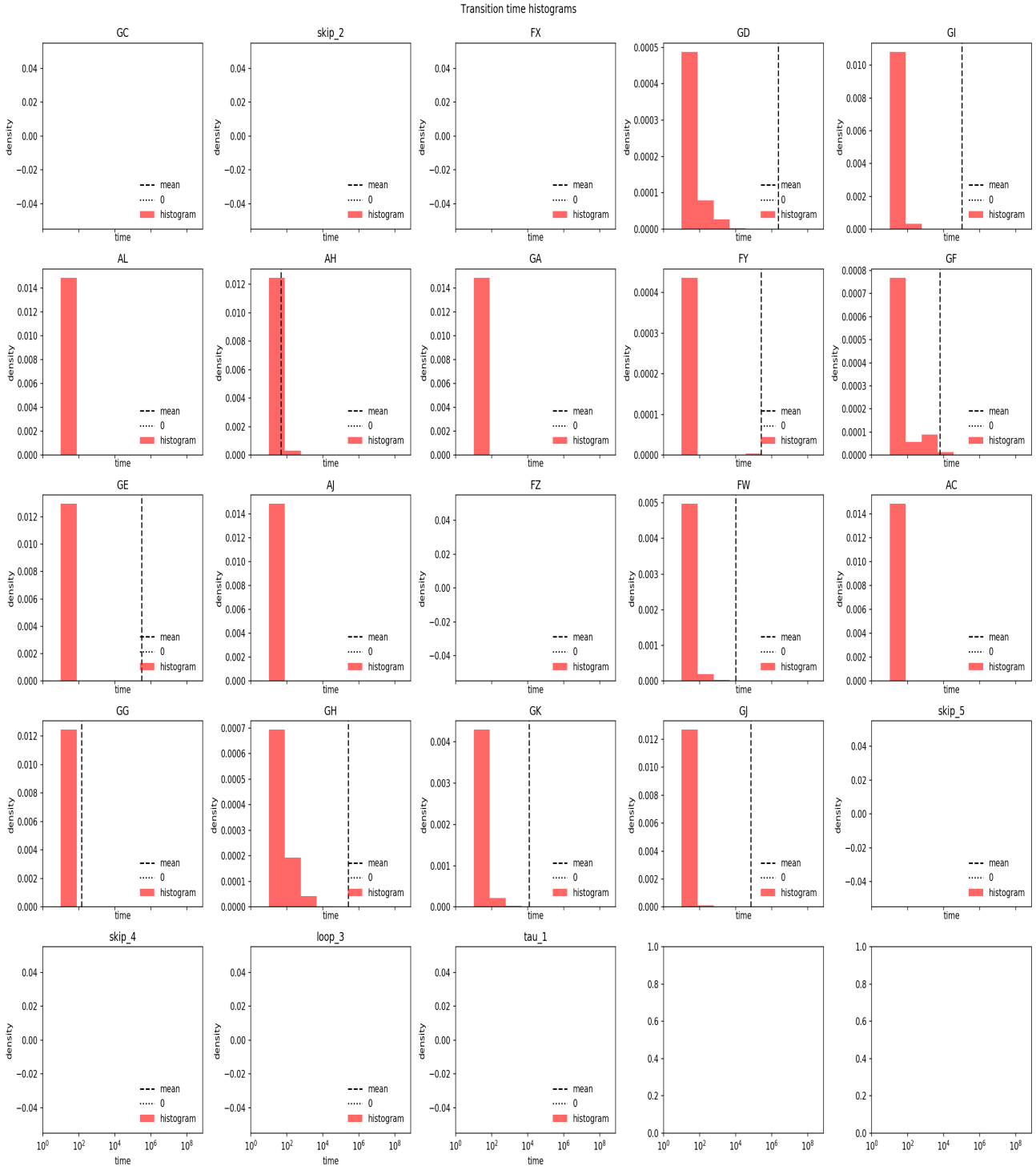


Figure 24: The histograms of the times taken to cross a particular transition.

	Transition Frequency	Minimum time	Maximum time	Mean time	Transition rate lower bound	Transition rate upper bound	$R\hat{2}$	$P(\lambda) = 0$	Passes Anderson-Darling test at 15pc significance
GD	0.160730	0.0	38709540.0	2.285669e+06	3.921625e-07	4.837298e-07	0.896204	5.676208e-09	False
GI	0.040952	0.0	56154060.0	1.024617e+05	9.577307e-06	9.961701e-06	0.861927	2.915160e-21	False
AL	0.003360	0.0	60.0	7.783019e-01	1.197801e+00	1.374465e+00	0.987510	8.857287e-04	False
AH	0.017225	0.0	660.0	4.833678e+01	2.008068e-02	2.133706e-02	0.918069	0.000000e+00	False
GA	0.006522	0.0	60.0	5.832321e-01	1.631691e+00	1.800905e+00	0.990554	6.124933e-05	False
FY	0.000135	60.0	4144380.0	2.632624e+05	2.498116e-06	5.106466e-06	0.883680	3.504739e-02	False
GF	0.050145	0.0	523080.0	6.223970e+03	1.579705e-04	1.636892e-04	0.945225	2.352844e-87	False
GE	0.166607	0.0	55454520.0	3.030145e+05	3.271251e-06	3.335693e-06	0.942075	1.335924e-125	False
AJ	0.008876	0.0	60.0	7.767857e-01	1.234189e+00	1.343099e+00	0.987542	6.630396e-08	False
FW	0.159221	0.0	36829140.0	9.923430e+03	9.986594e-05	1.018788e-04	0.985319	3.141320e-11	False
AC	0.011721	0.0	60.0	4.665314e-01	2.066720e+00	2.224523e+00	0.992401	1.555460e-06	False
GG	0.039462	0.0	86580.0	1.416709e+02	6.924070e-03	7.207278e-03	0.987916	9.574461e-17	False
GH	0.053668	0.0	60647640.0	2.611327e+05	3.767424e-06	3.899177e-06	0.949239	4.005799e-55	False
GK	0.159363	0.0	26935860.0	1.190325e+04	8.325603e-05	8.493336e-05	0.984295	2.511043e-16	False
GJ	0.122009	0.0	75716940.0	6.963040e+04	1.421205e-05	1.453976e-05	0.972589	7.245720e-33	False

Activities The CDF plots show that that while the fits visually agree quite well with the data $R^2 > 0.88$, the distributions disagree with the Anderson-Darling null hypothesis (at a 15% significance level) that the data was sampled from an exponential distribution. The fact that all tests rejected the null hypothesis at such a low significance level does not entirely rule out the possibility of memorylessness - it should be noted that all these tests were carried out independently and the introduction of distributions where memorylessness does not hold may cause assumptions to fail further on.

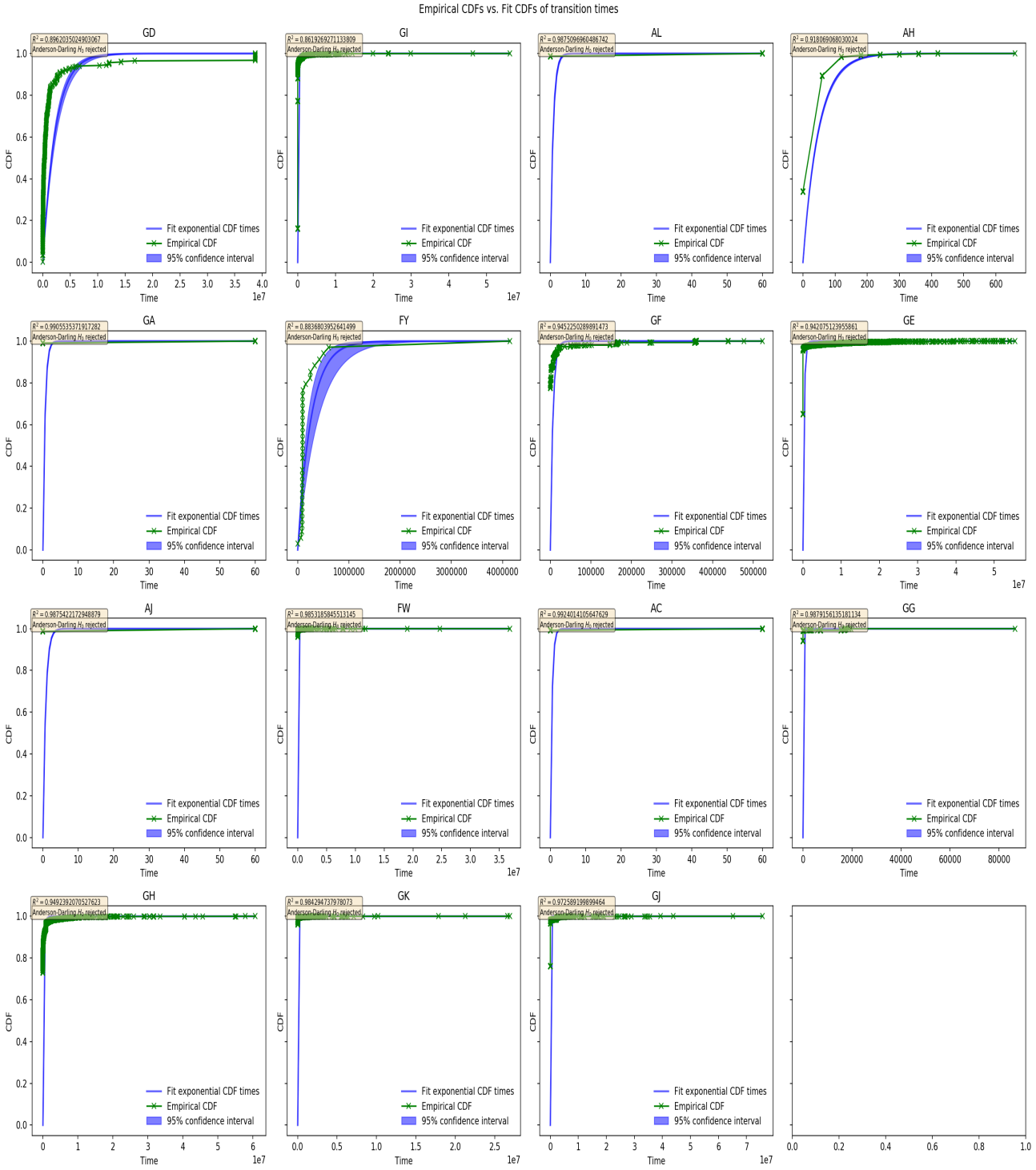


Figure 25: The empirical CDFs of the times taken to cross a particular transition.

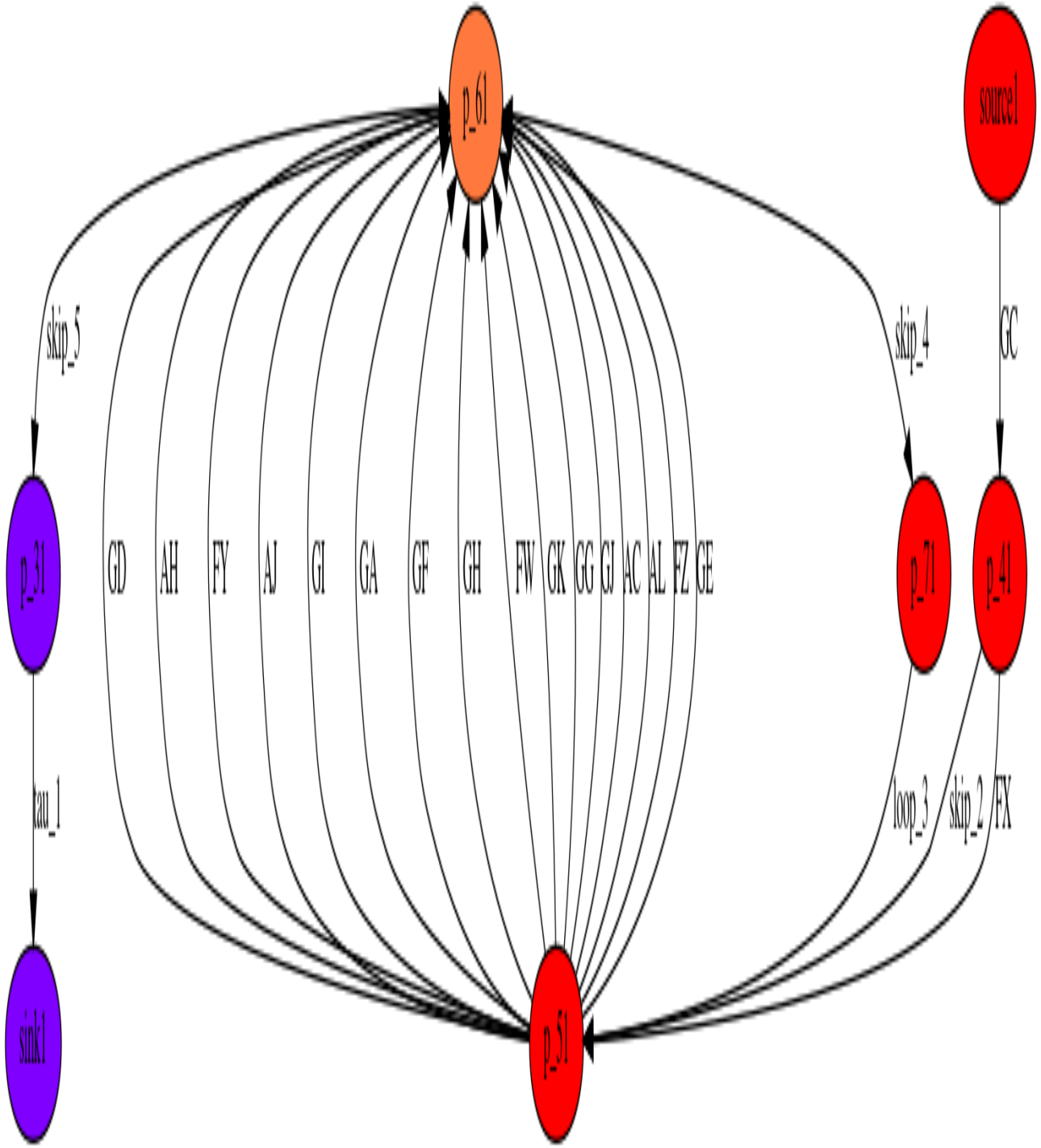
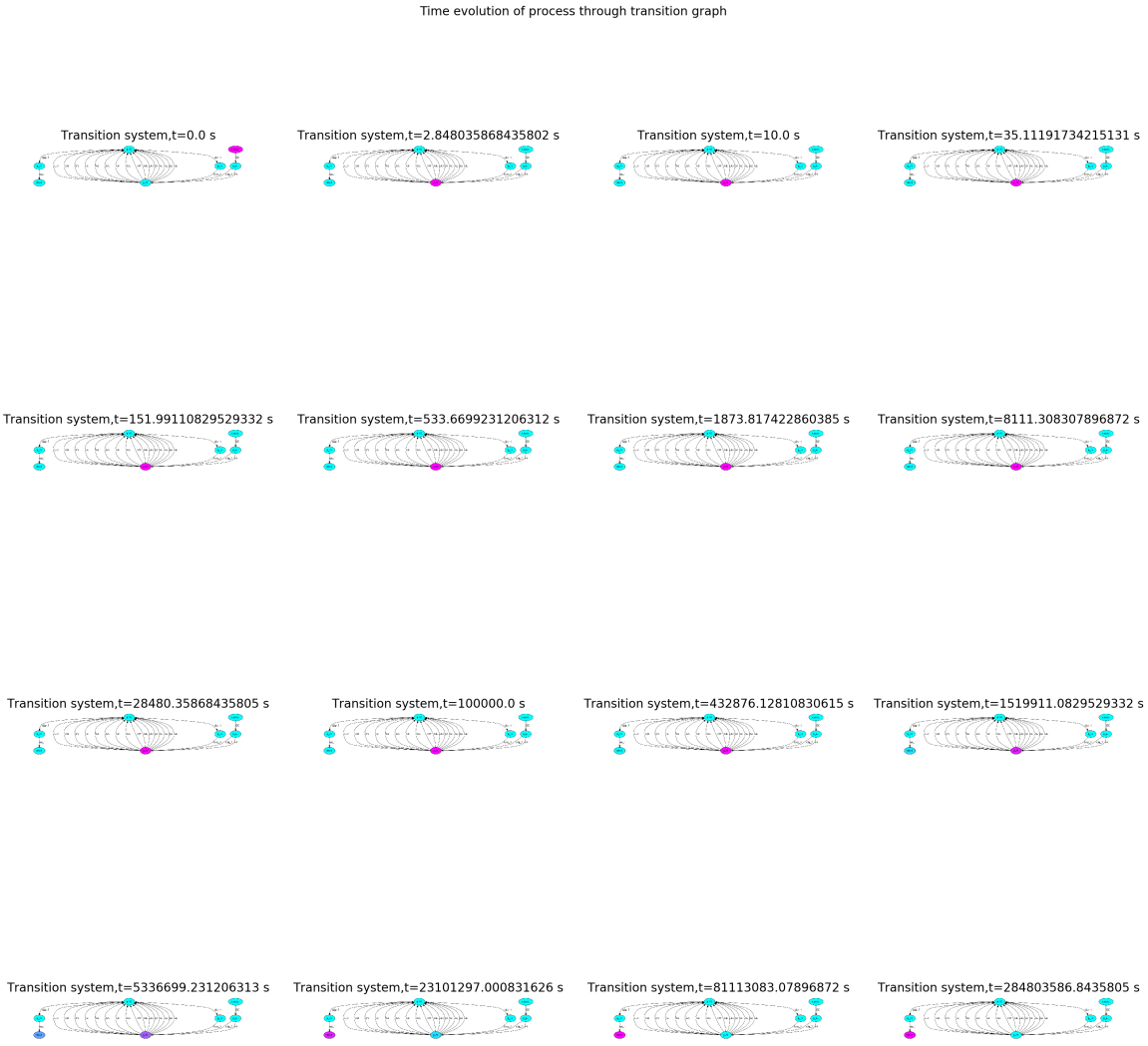


Figure 26: A reachability graph coloured by the mean hitting time it would take to get from this state to the end state `sink1`. Denoted $E_i[T_A] = k_i, A$ in the theory section, the more blue areas are close to 0 and the more red areas are close to the maximum of $E_i[T_A]$.

Modelled time-evolution The above diagram shows the transition system of the workflow net above. All traces pass through the series of transitions between `p_51` and `p_61`. The minimum rate for the set of transitions is the sum of the transition rates as shown in the methods section.



variant	Count	Fitness	Path Probability	Sum of mean act. times	se(Mean act. times)	Mean Case Duration	P(Mean act. times = Mean Case Duration)
GC,GD,GE,GJ,GK,FW	14223	1	0.0795818	2.67746e+06	1.22333e+08	203.544	0
GC,GD,GE,GJ,FW,GK	14222	1	0.0795818	2.67746e+06	1.22333e+08	245.113	0
GC,GD,GE,GF,GG,GH,GI,FW,GK	2688	1	0.0472778	2.97749e+06	1.51607e+08	292.121	0
GC,GD,GE,GF,GG,GH,GI,GK,FW	2628	1	0.0472778	2.97749e+06	1.51607e+08	263.699	0
GC,GD,GE,GF,GG,GH,AH,GH,GI,FW,GK	485	1	0.0334108	3.23841e+06	1.8197e+08	1.57693e+06	2.45831e-12
GC,GD,GE,GF,GG,GH,AH,GH,GI,GK,FW	418	1	0.0334108	3.23841e+06	1.8197e+08	1.55043e+06	5.74871e-13
GC,GD,GE,GF,GG,GH,GI,AC,GA,GK,FW	382	1	0.0334108	2.97749e+06	1.51607e+08	579383	1.98755e-87
GC,GD,GE,GF,GG,GH,GI,AC,GA,FW,GK	338	1	0.0334108	2.97749e+06	1.51607e+08	754941	2.1069e-38
GC,GD,GE,GF,GG,GH,GI,AC,FW,GK	270	1	0.039744	2.97749e+06	1.51607e+08	19157.8	0
GC,GD,GE,AJ,GE,GJ,GK,FW	267	1	0.0562397	2.98018e+06	1.84405e+08	6.16881e+06	1.66722e-08

Variant analysis Above is a graph of the 9 most common variants in the log. The null hypothesis that the mean variant time is equal to the sum of the mean activity times is rejected at the 95% confidence level by all process variants. Including more data from the less frequent variants in the $[GC, \ldots]$ chain results in the hypotheses of a memoryless process being rejected.

6 Discussion

6.1 Modelling notes

One useful factor about this formulation is that it has the capacity to be very simple and very scalable. Finding a workflow net that is simple ([3])s definition of simplicity can be done by performing some data preprocessing. A simple model employs a low amount of states in the transition graph to describe a majority of the activity in the process. It would be the transitions that make up the majority of the behaviour. In addition the IM_D algorithms tend to be quite efficient in terms of memory and running time due to the fact that they they rarely have to to look up in the log [8] . Time evolution calculations can be computationally intensive - ($O(n^3)$ calculations in the worst case scenario to solve a system of equations in n unknowns - but the worse case scenario can be approximated through approximate solutions such as least squares solutions, or even solved exactly and efficiently where the structure of the process results in a particular structure in the matrix. Certain Continuous-Time Markov chains follow particular known matrix structures and tend to be sparse. For example, the matrix density in the biggest case run here being no more than 0.25.

One particular obstacle found in the implementation of the model was that of numerical underflow. While this was dealt with by using higher precision arithmetic it would be advisable to anyone attempting build a transition matrix to use the log domain of the probabilities to evaluate sums of probabilities and possibly even to do the matrix calculations.

The structure of many processes could be approximated in a Markov sense but only in so far as the process is memoryless in general. It is possible that processes may be completed in a probabilistic sense - and that may also be a route for process mining in general.

6.2 Noise detection

Noise detection is a part of process mining that has been partly implemented in `PM4PY` at time of writing. Without a notion of being able to detect traces that are a product of noise, it is on the analyst to determine in the context of the analysis what traces count as noise. Without a noise-free log to work with and compare with of the same process, this can be approximated by taking frequent events more seriously than infrequent events under the assumption that correct behaviour is the most common. Using the language of sequence alignment, using metrics such as edit distance, Levenshtein distance or Hamming distance on traces and variants thereof can be a way to denoise traces.

For this analysis, the decreasing factor and percentage of data kept must be parameterized in a way that is most convenient for the analyst in the context of the use of the process model. Ideally a hyperparameter search could be done to find a optimal clean dataset that sense of optimality must be decided by the analyst themselves. In the context of trying to predict the time evolution of processes, The ability to recombine noisy variants in a noisy event log dataset back into the the variants they are most likely to represent would result in having more information on the distribution of each transition.

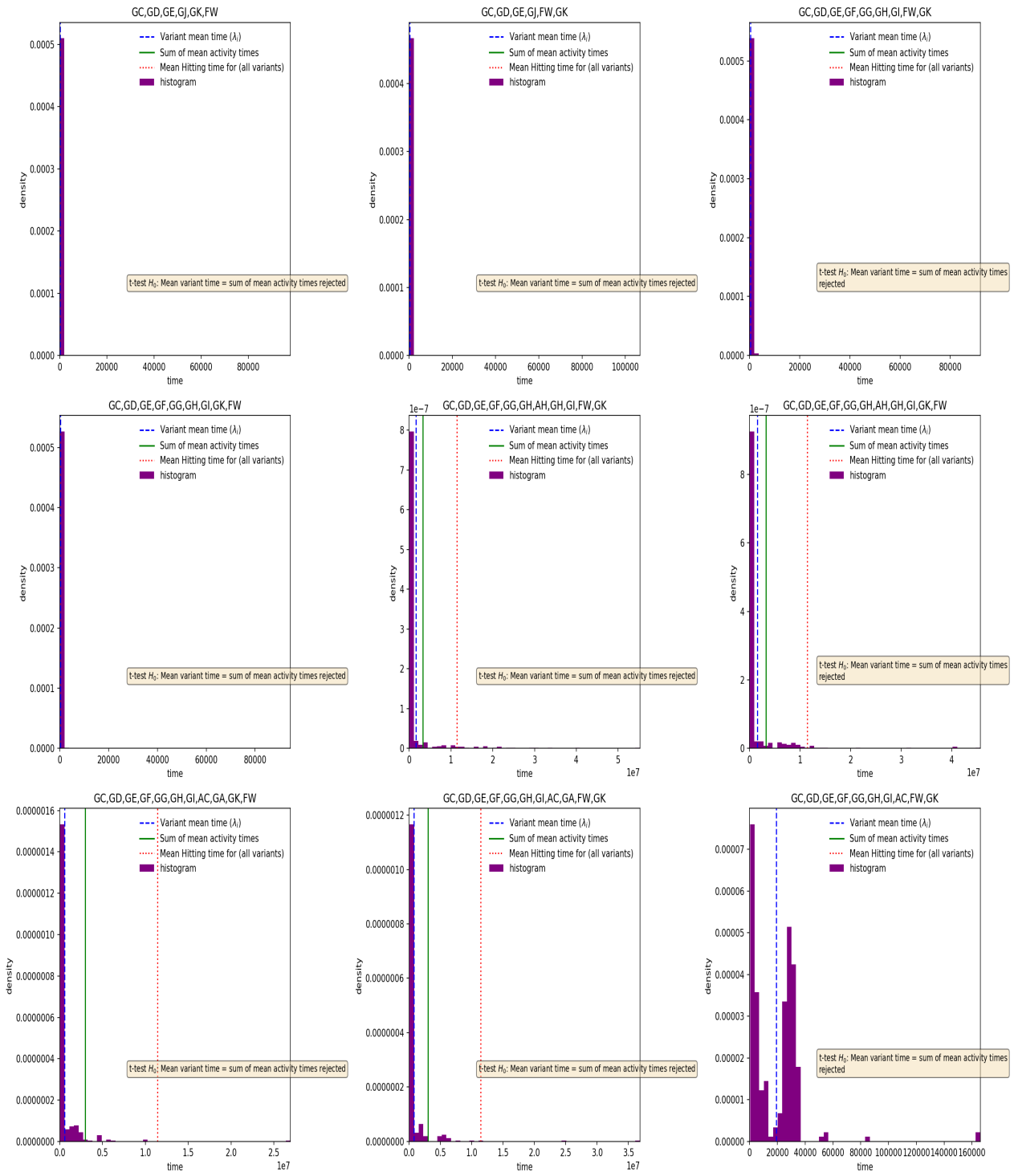


Figure 27: Histograms of the 9 most frequent variants of traces in the data. Various landmark times are shown above.

6.3 Completion time

The general question of how long a process takes has been tackled before in the literature. A core question of the research done here was to see if the activities in a process log in an event log could be modelled like the states in a Markov Chain.

The Markov Property as a whole depends on a idea of memorylessness. Memorylessness is verified in this work by testing that the probability of going from one place in the petri net to another does not depend on any previous travel through the workflow net before it. However on inspection and investigation it is likely that a strong notion of memorylessness may be too stringent for general use and may result in incorrect estimations of time taken to complete the process. It should be noted that the continuous time Markov chain model for transition systems and petri nets is a very useful and natural generalization of a petri net, and a simple representation of a stochastic petri net, but it should be with strong caveats bearing in mind that a process - debatably by design - have a sense of memory to it. Constructs built with petri nets can represent this memory but the degree to which this memory is represented in the probabilistic model of that process is difficult question. One possible direction for this work to go in in the future would be to investigate the modelling potential of n^{th} order Markov chains ie. Markov chains with memory on transition systems. However knowing that transition systems have a state explosion problem in the event of high numbers of parallel processes. Fitting a n^{th} order Markov chains requires a fitting on "bigrams (2nd order)" or "trigrams (3rd order)" of states and will require the amount of states to rise exponentially. These models could become very intractable very quickly especially for larger processes and logs.

6.4 Fit quality

Use of the exponential distribution in this work is a sticking point for the work as a whole however this is a core approximation of of the memoryless stochastic petri nets structure. Without the memoryless firing rate assumption a more sophisticated way of determining which transitions in a workflow net actually fire must be found by making assumptions about firing known as firing rules. This adds complexity to the model. Analysis of these stochastic petri nets called generalized stochastic petri nets tends to use queuing theory and requires in enforcing a firing policy on the petri net. Processes tend to follow unspecified distributions in general, but if reasonable hypotheses about the distribution of the processes can be generated this model can work very well.

Another option is to fit a model using standard machine learning techniques and regression techniques to the event log. This would involve creating a model that would be fit using $|t| - 1$ dummy variables to represent the $|t|$ transitions. transitions as dummy variables. This could allow the finishing time of the process to be approximated using classical regression techniques. This could be further refined by using metadata on the activities to find correlations between them total finishing time. Another alternative would be to use a prefix log: a version of the event log that contains cumulative information on each trace as it occurs. A prefix log would be an event log that for each activity added onto a case in an event log, the prefix log would create a new subcase of that case that would contain all the activities added to the trace cumulatively. This prefix log would get very large very quickly but this would allow for machine learning algorithms to fit on in-progress processes. This has been done before and has been done in ([16]) using support vector regression and neural networks [17]. In addition much success has been had in using machine learning techniques such as decision trees to perform root cause analysis - this has been implemented in `PM4PY` [3] . on the process. Features such as long transition times and sources of bottlenecks in a process can be found using supervised learning techniques or even unsupervised techniques to cluster the traces and predict process times by correlating the the meta data in those processes with the likelihood of a certain process having a high or anomalous time. Furthermore, work has been done in using social network graphs to investigate if people or specific agents such as machines are correlated with particular states of processes.

While the examples given in this project have shown this method is a poor fit for the data, there are a number of alternative formulations of the process that can allow for time prediction, even still staying within the realm of Markovian processes. It may also be possible to partition workflow nets in a way to exploit the concept of memory. By delegating the concept of memory to a suitable petri

net construct, chains of memoryless places and transitions could replace single problematic transitions. This would increase the amount of states and transitions in each model but would theoretically allow a degree of memory to be dealt with in a model.

One other factor that resulted in the exponential distribution being a poor fit for many of the transitions is that some of the data had minimum values greater than zero. The usual memoryless property of the exponential distribution relies on the use of only the parameter *lambda* inside the exponential function and does not allow for a location parameter to be used: ie. Some implementations of statistical software, `scipy` in particular, allow fitting to exponential distributions with a location parameter by default. This would look like $P(t; \lambda, t_0) = \lambda e^{-\lambda(t+t_0)}$ While this would improve the fit of the model especially in the tails, this parameterisation violates the memorylessness property in the regime where $t < t_0$.

It may also be possible to use a Markovian representation of the model but translating the process to a time inhomogeneous regime to approximate the location parameterisation in such a way that $e^{\lambda(t)t} \approx e^{\lambda(t+t_0)}$. This would end up being $\lambda(t) \approx -\lambda(1 + \frac{t_0}{t})$. This would increase the complexity of the model by a large margin by having to find an extra parameter for each transition but could help the fit by alllowing the transition matrix as a whole to change over time.

6.5 Memory testing

Multivariate testing for memorylessness in the process as a whole may add some additional evidence to strengthen a hypothesis that a process is memorylessness. In this case univariate tests suffice due to the sensitivity of the the memorylessness condition. Any evidence of memory in even one transition in the process fundamentally violates the memorylessness conditin and one rejected null hypothesis implies a wholesale rejection of memorylessness. Worth noting is how how much of the data seems to follow a exponential distribution to the human eye but fails the Anderson-Darling tests. These tests are very sensitive in the tail end of a distribution and this is notably where less data was available - the very nature of studying the tail regions of a distribution necessitates an abundance of data. This would lead to more data and more accurate evaluation of the distributions of the transitions. With more data it may be possible to actually surmise that the the distributions studied here in were in fact exponential but simply did not have enough data to find rare behaviour.

7 Conclusion

As can be seen from the results above the process mining segment of the modelling process tends to fit the data quite well and in some cases the IM_D algorithms, actually achieve full fitness to even large datasets. This is by design due to the process tree construction but as mentioned before this causes a lot of redundancies in the models found. These redundancies must be carefully trimmed down dude with careful processing of the data. It was the case that in both of these data sets both the decreasing factor method and the removal of infrequent variants improved the fit in different but important ways. Using one method over the other tended to to create models that were more general in a sense but also less succinctly captured the mainstream process. Simply just removing infrequent variants from the data would create a smaller more simple model but would sometimes lead to models overfit and forced structures on what ended up being unstructured sequences of events, as was seen working with the energy trading metadata with the causal chain when only fit to the top 50% most frequent variants in the data. In contrast relying on only the decreasing factor method would result in workflow nets that would start and end simply but use infrequent behaviour between the start and end points, keeping the model general but including many infrequent events which may turn out to be noise.

8 References

References

[1] Wil van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malu Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn F. van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul, Frank van Geffen, Sukriti Goel, Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maggi, Donato Malerba, Ronny S. Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid R. Motahari-Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Wynn. Process mining manifesto. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, *Business Process Management Workshops*, pages 169–194, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[2] W.M.P. van der Aalst. Structural characterizations of sound workflow nets, 1996.

[3] Alessandro Berti, Sebastiaan J. van Zelst, and Wil van der Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science, 2019.

[4] H. Pishro-Nik. Introduction to probability, statistics, and random processes, 2014.

[5] M. Ajmone Marsan. Stochastic petri nets: An elementary introduction. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1989*, pages 1–29, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[6] Wil van der Aalst. *Process Mining*. Springer Berlin Heidelberg, 2016.

[7] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs - a constructive approach. In José-Manuel Colom and Jörg Desel, editors, *Application and Theory of Petri Nets and Concurrency*, pages 311–329, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Scalable process discovery with guarantees. In Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 85–101, Cham, 2015. Springer International Publishing.

[9] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *Proceedings of the 26th International Conference on Applications and Theory of Petri Nets*, ICATPN'05, pages 444–454, Berlin, Heidelberg, 2005. Springer-Verlag.

[10] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[11] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed <today>].

[12] Mohd Anuaruddin Bin Ahmadon and Shingo Yamaguchi. Convertibility and conversion algorithm of well-structured workflow net to process tree. In *2013 First International Symposium on Computing and Networking*. IEEE, dec 2013.

[13] M Taboga. Lectures on probability and statistics, 2010.

[14] Richard Durrett. *Essentials of Stochastic Processes*. Springer New York, 2012.

[15] Federico J. O'Reilly and Michael A. Stephens. Characterizations and goodness of fit tests. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(3):353–360, 1982.

[16] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Time and activity sequence prediction of business process instances. *Computing*, 100(9):1005–1031, Feb 2018.

[17] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In *International Conference on Advanced Information Systems Engineering*, pages 477–492. Springer, 2017.