Deriving Coloured Generalised Stochastic Petri Net Performance Models from High-Precision Location Tracking Data

Nikolas Anastasiou
Department of Computing
Imperial College London
South Kensington Campus
London SW7 2AZ
na405@doc.ic.ac.uk

William Knottenbelt
Department of Computing
Imperial College London
South Kensington Campus
London SW7 2AZ
wjk@doc.ic.ac.uk

ABSTRACT

Stochastic performance models are widely used to analyse systems that involve the flow and processing of customers and resources. However, model formulation and parameterisation are traditionally manual and thus expensive, intrusive and error-prone. Our earlier work has demonstrated the feasibility of automated performance model construction from location tracking data. In particular, we presented a methodology based on a four-stage data processing pipeline, which automatically constructs *Generalised Stochastic Petri Net* (GSPN) performance models from an input dataset of raw location tracking traces. This pipeline was enhanced with a presence-based synchronisation detection mechanism.

In this paper we introduce *Coloured Generalised Stochastic Petri Nets* (CGSPNs) into our methodology to provide support for multiple customer classes and service cycles. Distinct token types are used to model customers of different classes, while Johnson's algorithm for enumerating elementary cycles in a directed graph is employed to detect service cycles. Coloured tokens are also used to enforce accurate customer routing after the completion of a service cycle. We evaluate these extensions and their integration into the methodology via a case study of a simplified model of an Accident and Emergency (A&E) department. The case study is based on synthetic location tracking data, generated using an extended version of the LocTrackJINQS location-aware queueing network simulator.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modelling Techniques; I.6.5. [**Model Development**]: Modelling Methodologies

Keywords

Location Tracking, Performance Modelling, Data Mining, Coloured Generalised Stochastic Petri Nets

1. INTRODUCTION

Complex customer-processing systems are frequently encountered in everyday life. Such systems, e.g. airports, hospitals and factory assembly lines, are subject to stringent Quality of Service (QoS) requirements. Failure to comply with these requirements can have severe financial implications and may lead to high profile fiascos, such as the opening of Heathrow Terminal 5[1] and the unacceptably long waiting times of patients in Accident and Emergency departments in the UK[2]. Accurate performance models of these systems can be used as a predictive tool to ensure the provision of adequate QoS under foreseeable workloads. Unfortunately, the availability and quality of such models are currently limited due to the associated construction cost, both in terms of time and money, and the high degree of manual intervention required, in terms of data collection, model construction and model parameterisation.

The gradual realisation of the Internet of Things and the increasing adoption of Real Time Location Systems (RTLSs) have led to the availability of low-level data which describes the movements of customers and/or resources in a given system. In this context, our earlier work demonstrated the feasibility of automated performance model construction from location tracking data [2]. In particular, we presented a methodology, based on a four-stage data processing pipeline, which automatically constructs *Generalised Stochastic Petri Net* (GSPN) [15, 4] performance models from raw location tracking traces. The developed methodology proved its ability to capture the stochastic features and abstract structure of customer-processing systems which satisfy the following assumptions: static service centres, random service discipline, single customer class, single-server service semantics and probabilistic customer routing. The resulting models can be visualised using PIPE2, the Platform Independent Petri net Editor [5, 7].

In subsequent research, we developed a presence-based synchronisation detection mechanism [3]. This determines whether the processing of customers at each service area depends on the presence of a certain number of customers at other service areas. If synchronisation is detected, then the extracted service time samples are adjusted accordingly.

[1]`http://www.computerweekly.com/news/2240086013/British-Airways-reveals-what-went-wrong-with-Terminal-5`
[2]`http://www.guardian.co.uk/society/2012/may/31/patients-waiting-four-hours-2004`

We now introduce *Coloured Generalised Stochastic Petri Nets* (CGSPNs) [14] into our methodology in order to address some of its limitations. In particular we provide support for multiple customer classes and allow customers to be routed according to *service cycles*. Different token types (represented by distinct colours) are used to model customers of different classes. Johnson's algorithm for enumerating elementary cycles in a directed graph [11] is used to identify service cycles. Coloured tokens are also used to enforce accurate customer routing after the completion of a service cycle. Together these extensions allow for more realistic modelling of complex customer-processing systems.

The rest of this paper is organised as follows. Section 2 presents previous research related to our work. Section 3 provides an overview of our existing and modified methodologies. Section 4 presents a case study to examine the accuracy and effectiveness of the modified approach. This uses synthetic location tracking data generated using an extended version of LocTrackJINQS [9]. The paper concludes with a summary of the results and a discussion on future work.

2. RELATED WORK

Previous research conducted by Kounev et *al.* examined the automatic construction of performance models in a different context [12]. Their work presents an approach to automatically extract the performance model of an *Enterprise Data Fabric* (EDF) in the form of a *Queueing Petri Net* (QPN) [4]. An EDF is a distributed enterprise middleware, located between the application and the host network, and it is used to allocate and manage data and resources across multiple, physically separate, hardware nodes. The authors parameterise the model using readily available monitoring data which are provided by the EDF. This approach is implemented as part of the simulation-based tool Jewel whose purpose is the automated performance prediction and capacity planning for EDFs.

Another research endeavour in the field of automatic construction of GSPN models has been made by Xue et *al.* [17]. Their methodology is applicable to *Flexible Manufacturing Systems* (FMSs) and it is implemented by the FMSPet program. An FMS consists of two components: the actual manufacturing system, i.e. an assembly line, and a controller which allows the system to react to various changes. FMSPet requires as input the description of the FMS, specified by an input language called FMSDL (presented in [17]), and produces a GSPN model for the underlying system. However, the methodology does not extract the model from data; the FMS must be explicitly defined in the input file.

Earlier work conducted by Horng et *al.* [10] is the most closely related to ours. The authors present a methodology designed to infer simple Queueing Network performance models from high-precision location tracking data. The constructed models capture the structure of certain restricted classes of underlying system – specified in terms of the routing probabilities of the customer flow – and the service time distributions of its service areas. The inter-arrival time distribution of customers at service centres is also inferred. Unlike in our present approach, however, there is no ability to infer automatically the locations and radii of service areas, the existence of presence-based synchronisation between service areas or the presence of service cycles.

3. INFERRING PNPMS FROM HIGH-PRECISION LOCATION TRACKING DATA

3.1 Existing Data Processing Pipeline

Here, we provide an overview of our prior work [2, 3] since some basic understanding of the existing methodology is required in order to introduce new features. A high-level description of the original data processing pipeline is shown in Figure 1. The pipeline takes as input a set of raw location tracking data, which describes the customer flow in a customer-processing system, and outputs a Petri Net performance model (PNPM) of the underlying system.

The first stage of the pipeline prepares the raw location traces[3] for processing by the subsequent stages. In particular, it converts the input data into a standardised format and separates it into customer paths, one for each recorded customer. A customer path consists of the location updates associated with a particular customer's movements throughout its stay in the system. A speed-based filter is then applied on each customer path in order to remove location updates that imply infeasibly rapid movement.

The second stage consists of three phases which infer the locations and radii of stationary service areas in the system. It operates under the assumption that customers stop or slow down while receiving service. The three phases are: speed filtering, density filtering and the application of the DBSCAN clustering algorithm [8]. The speed and density filters are applied on each customer path and aim to identify the multiset of positions (two-dimensional Cartesian coordinates) where the customer was stationary or moving at 'low' speed. These multisets – one for each customer path – are aggregated into a single dataset on which the DBSCAN algorithm is applied. The centroids of the produced clusters are used to approximate the locations of the system's service areas. The radius of a service area is conservatively approximated as 110% of the 95th percentile of the distance between the corresponding cluster's centroid and each of its contained points.

Stage three constructs the basic structure of the derived PNPM, beginning with the places and transitions required to represent the flow of customers in the system. We distinguish between two types of places: places associated with service areas (inferred from stage two) and places associated with customer movement between service areas. We call these server and travel places respectively. Similarly, we distinguish between two types of transitions: those which connect server to travel places and those connecting travel to server places. The latter are called travelling time transitions and the former service area service time transitions. These transitions are not currently parameterised; they are replaced during the next stage by a GSPN subnet that accurately reflects the distribution of the relevant time delays. In preparation for this, we compute response time samples – for each customer – inside a service area broken into waiting time and service time. Samples of the time required by each customer to transit between two service areas are also computed (travelling time samples). Then, the presence-based synchronisation detection mechanism [3] is applied. Whenever synchronisation involving the processing of customers

[3]A typical location update reading from an RTLS is a tuple of the form (`tagName, type, time, x, y, z, stderr`).

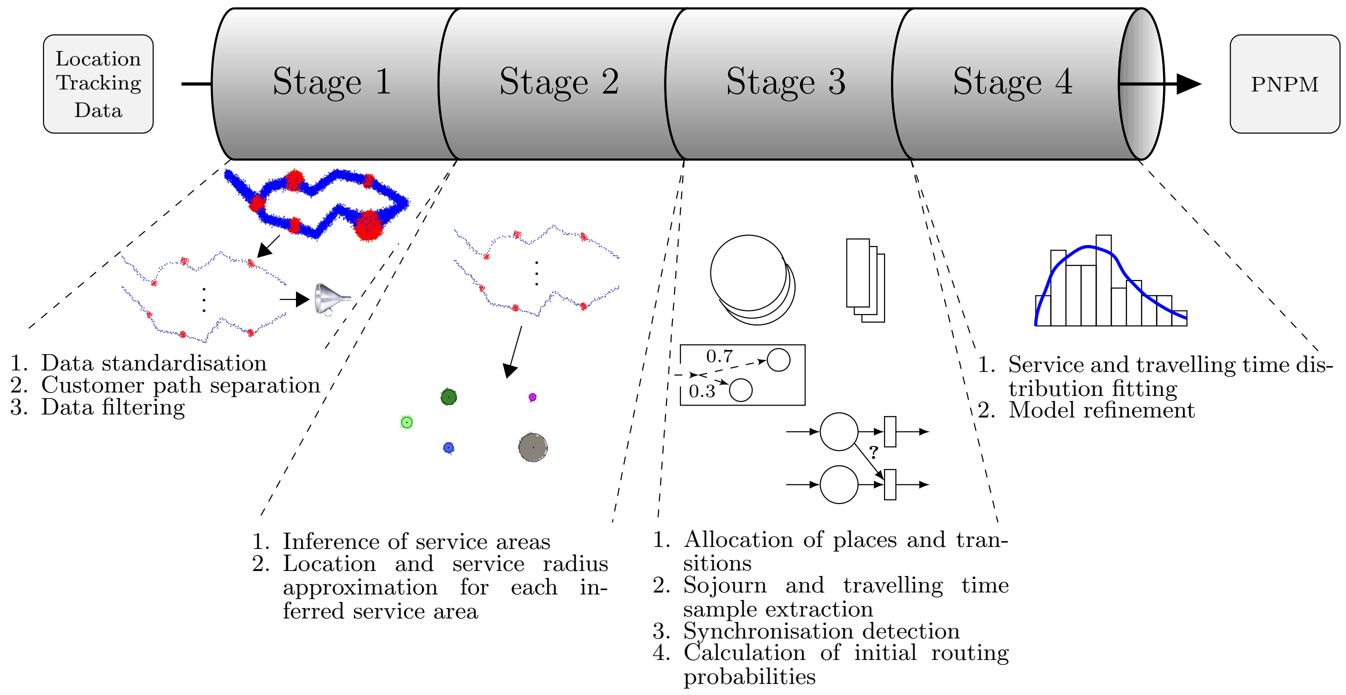

Location Tracking Data
Stage 1
Stage 2
Stage 3
Stage 4
PNPM
1. Data standardisation
2. Customer path separation
3. Data filtering
1. Inference of service areas
2. Location and service radius approximation for each inferred service area
0.7
0.3
?
1. Allocation of places and transitions
2. Sojourn and travelling time sample extraction
3. Synchronisation detection
4. Calculation of initial routing probabilities
1. Service and travelling time distribution fitting
2. Model refinement

Figure 1: The four-stage data processing pipeline as in [3].

at a particular service area is detected, the corresponding service time samples of those customers are adjusted to take into account the proportion of time during which the synchronisation condition(s) are satisfied[4]. Finally, a counting mechanism calculates the initial and inter-routing probabilities of the customer flow.

The fourth stage of the pipeline replaces the service area service time transitions and travelling time transitions with GSPN subnets that reflect the distributions of the corresponding service and travelling time samples computed in stage three. For each set of these time samples we fit several hyper-Erlang distributions (HErDs) using the G-FIT tool [16]. Each GSPN subnet is constructed in such a way that it reflects the best-fit HErD which is selected using the Akaike Information Criterion (AIC) [1].

3.2 Modelling Multiple Customer Classes

The first extension introduces CGSPNs into the data processing pipeline in order to support multiple customer classes. The main assumption here is that the class of each customer in the system is known and provided through the customer's associated location updates, e.g. the `type` field of a typical location update contains the category the monitoring tag belongs to, or by a static mapping of each customer's unique identifier (`tagName`) to the class it belongs to. Here we focus on the third stage of the pipeline, especially on the way we form the initial structure of the PNPM.

In a CGSPN, each transition can support many firing modes, i.e. a transition can be enabled under several different markings. For example, consider the CGSPN depicted in Figure 2 where p_1 is marked with three token types: two blue, two red and one black. The transition t_1 may require two blue tokens to be enabled, or one red, or one black, or even a combination of distinct types, e.g. one blue, one red and one black. The set of markings under which t_1 is

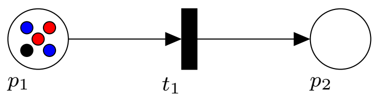


Figure 2: A simple CGSPN with three token classes.

enabled must be specified so that the dynamic behaviour of the model accurately reflects the underlying system's behaviour. We note that a different firing rate or weight can be associated with each supported firing mode of a timed or immediate transition respectively.

To facilitate the unambiguous visualisation of a transition's firing modes we use a different transition for each firing mode that we wish to support. That is, if we consider the latter example, assuming that t_1 has three firing modes (see Table 1) we use three transitions; this is shown in Figure 4.

Section 3.1 presented an overview of the existing pipeline, which included the various tasks performed during the third stage. When dealing with multiple customer classes the process of place construction remains the same. Service area service time transitions are created in a similar manner as before but instead of having one such transition we now have multiple transitions: one for each class of customers that was processed by the corresponding service area. The same principle applies for the construction of travelling time transitions. The total number of customer classes is obtained during the computation of the response and travelling time samples by simply counting the distinct classes observed. The operation of this process requires no modifications; how-

[4]We assume that service progresses only when the synchronisation condition(s) are met.

Mode	$I^-(p_1,t_1)$(red)	$I^-(p_1,t_1)$(blue)	$I^-(p_1,t_1)$(black)	$I^+(p_2,t_1)$(red)	$I^+(p_2,t_1)$(blue)	$I^+(p_2,t_1)$(black)
1	1	0	0	1	0	0
2	0	1	0	0	1	0
3	1	1	1	1	1	1

Table 1: The supported (three) firing modes of transition t_1 shown in Figure 2. $I^-(p_i,t_j)(c_k)$ denotes the number of tokens of colour c_k that must be present in p_i in order for t_j to be enabled. $I^+(p_i,t_j)(c_k)$ denotes the number of tokens of colour c_k that are created in p_i when t_j fires.

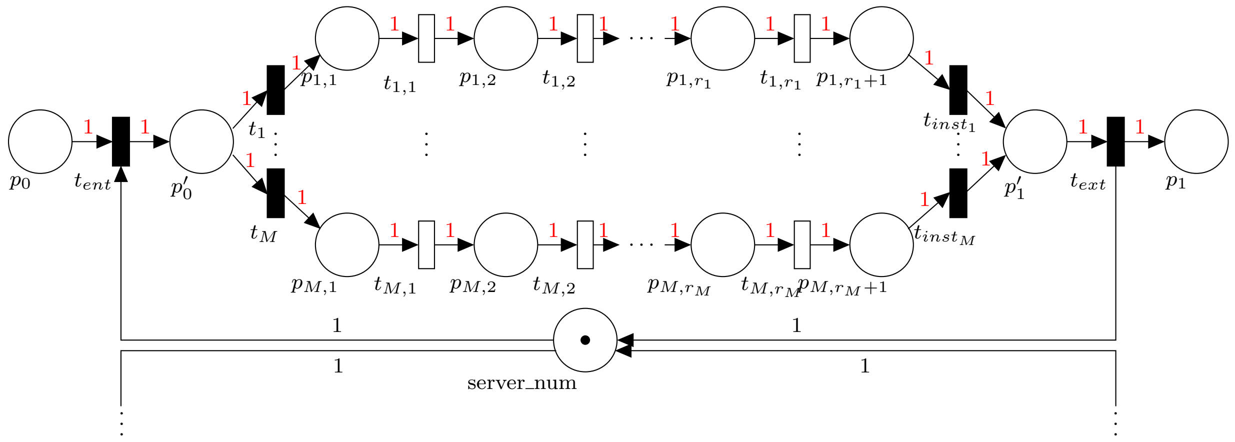


Figure 3: CGSPN subnet representation for service area service time transitions. The colour of tokens supported varies according to the customer class to which the net applies. The complementary place labelled as server_num is shared among subnets associated with the same service area. This allows only one token to be in a subnet at a time and thus prohibits parallel service of distinct customer classes (we assume single-server semantics). p_0 represents a service area and p_1 is associated with the movement of customers between p_0 and some other service area.

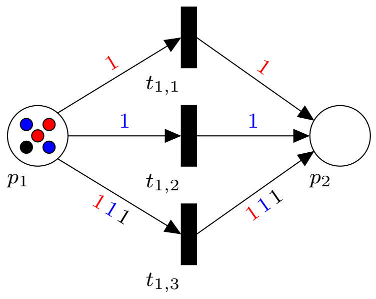


Figure 4: The CGSPN shown in Figure 2 with one transition for each different firing mode of t_1 (cf. Table 1).

ever, the computed service and travelling time samples are now separated into groups, one for each customer class.

The GSPN subnet reflecting the best-fit HErD of a particular group of time samples now becomes a CGSPN. The firing mode of each transition in the subnet is set according to the class of customers the group of time samples corresponds to. The general form of a CGSPN subnet used to replace service area service time transitions is shown in Figure 3. All such subnets that are associated with a particular service area share the same complementary place server_num. This place contains one token and ensures that no more than one token is allowed in a subnet simultaneously (thus prohibiting parallel service of customers of different classes). Subnets used to replace travelling time transitions are similarly structured but without the complementary place server_num.

3.3 Service Cycles and Customer Routing

The approach presented here assumes one class of customers. The extension to multiple customer classes is discussed in the next section.

Often, in customer-processing systems, routing is not probabilistic but deterministic with several phases of processing. The journey through these processing phases may involve repeated visits to the same service centres. In order to be able to model this kind of customer behaviour we introduce the concept of a *service cycle*. An example of such a system is shown in Figure 5 where the service cycle consists of Service Area A, B and C. Formally,

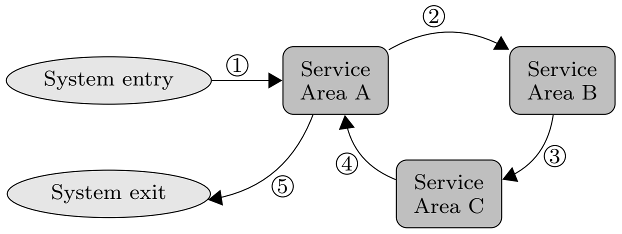


Figure 5: A customer-processing system which provides services to customers in a cyclic fashion. Arrows indicate the flow of customers in the system.

Definition 1. Consider a physical customer-processing system consisting of the set of service areas $S = \{S_i \mid i = 1,\ldots,N\}$, for some finite $N \in \mathbb{N}$. A finite sequence of *distinct* service areas from which the customers of the system obtained some service, is said to be a service cycle given that the following conditions hold:

1. If the customers initiate their service sequence at S_i, $S_i \in S$, they must also terminate it at S_i after service completion at S_k, $S_k \in S$ and $i \neq k$.
2. When the customers complete their service at S_i for the second time, i.e. after obtaining service at S_k, they do not repeat the same service sequence.

The service area which marks the initiation and termination of the service cycle is called the *head* of the service cycle.

If the existing methodology is applied to a set of location tracking data retrieved from the system depicted in Figure 5, under the assumption of one customer class, then the inferred GSPN model (non-parameterised version) is shown in Figure 6. Although the structure of the model resembles

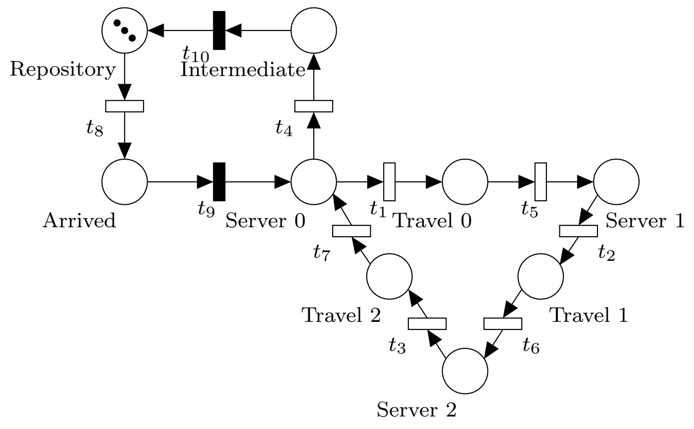


Figure 6: The GSPN model (non-parameterised version) for the system depicted in Figure 5 using our existing methodology. Places labelled as Server 0, Server 1 and Server 2 represent the service areas A, B and C respectively.

the underlying system, the model fails to capture the desired behaviour and routing of customers in the system. In particular, the second condition of Definition 1 can be possibly violated because of the race condition between transitions t_1 and t_4; when the place Server 0 is marked with at least one token t_1 and t_4 are simultaneously enabled. The transition to fire is selected probabilistically and the probability of each transition depends on their firing rates.

The key idea here is to identify the transitions involved in such race conditions and adjust their firing mode, i.e. change the supported token type, so that the race condition ceases to exist. The different token types are used to express the customers' "phases of service". That is, we distinguish between customers who have obtained service from each service area contained within the service cycle and customers who are about to enter it, i.e. obtain service from the head of the service cycle. We classify the former and latter groups of customers as serviced and unserviced respectively. The CGSPN model (non-parameterised version) that is obtained for the previous example when this modelling approach is applied, is shown in Figure 7.

To apply this approach, the system being modelled must first be examined for the presence of service cycles. To perform this task we obtain a directed graph G, $G = (V, E)$,

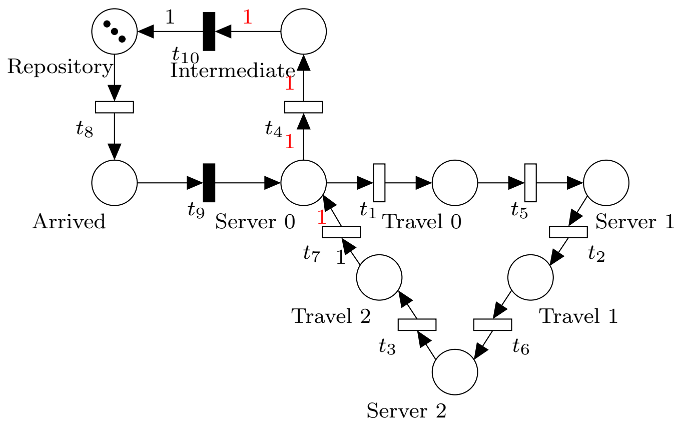


Figure 7: The CGSPN model (non-parameterised version) for the system depicted in Figure 5, which enables the distinction between serviced (red tokens) and unserviced (black tokens) customers. Where no arc inscription is explicitly shown one black (default) token is assumed.

which represents the flow of customers in the system. V denotes the set vertices of the graph and E the set of its edges. This graph is derived from the non-parameterised model by mapping each place in the model to a vertex and each transition – along with its incident arcs – to a directed edge (cf. Figure 8). We then use Johnson's algorithm [11]

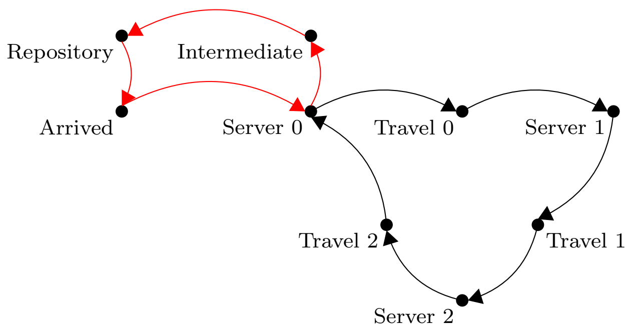


Figure 8: The directed graph produced for the Generalised Stochastic PNPM depicted in Figure 6. The paths consisting of black and red edges are two elementary cycles beginning at vertex Server 0.

to ascertain whether the derived graph contains elementary cycles. An elementary cycle is a sequence of vertices $c_v = (v_1 = v, v_2, \ldots, v_k = v)$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < k$, where no vertex, apart from the first and last, appears twice. For more information we refer the reader to a standard graph theory textbook, e.g. [6]. This algorithm requires as input the directed graph and outputs the sequence of vertices that form each identified elementary cycle. The direct application of the algorithm on G would always return at least one elementary cycle which will include the Repository place, e.g. see Figure 8. This place of the PNPM represents the entry/exit point of the underlying system and thus, is actually not part of any service cycle. To avoid such unwanted cycles to be detected by the algorithm we examine the subgraph of G induced by the set of vertices

Transition	Original Firing Mode		Modified Firing Mode	
t_4	$I^-(\text{Server } 0, t_4)(\text{black}) = 1$	$I^+(\text{Interm.}, t_4)(\text{black}) = 1$	$I^-(\text{Server } 0, t_4)(\text{red}) = 1$	$I^+(\text{Interm.}, t_4)(\text{red}) = 1$
t_7	$I^-(\text{Travel } 2, t_7)(\text{black}) = 1$	$I^+(\text{Server } 0, t_7)(\text{black}) = 1$	$I^-(\text{Travel } 2, t_7)(\text{black}) = 1$	$I^+(\text{Server } 0, t_7)(\text{red}) = 1$
t_{10}	$I^-(\text{Interm.}, t_{10})(\text{black}) = 1$	$I^+(\text{Rep.}, t_{10})(\text{black}) = 1$	$I^-(\text{Interm.}, t_{10})(\text{red}) = 1$	$I^+(\text{Rep.}, t_{10})(\text{black}) = 1$

Table 2: The required firing mode modifications, if any, for each transition of the PNPM shown in Figure 6 to enable the accurate representation of the underlying system's customer flow (Figure 7). Interm. and Rep. denote the places labelled as Intermediate and Repository respectively.

$V - \{\text{Repository}\}$ instead of G. Also, according to the following theorem we restrict the algorithm's search to begin only with vertices which have at least two *predecessors*[5] and at least two *successors*.

THEOREM 1. *Given a customer-processing system and its corresponding directed graph, a service cycle exists only if the vertex v_i which corresponds to the head of the service cycle S_i has at least two predecessors and at least two successors.*

The proof of Theorem 1 is shown in the Appendix. When the algorithm completes its search, it returns the ordered list of vertices V_i which constitute the elementary cycle – one list for each cycle i. For each such list we obtain the (ordered) list of the corresponding places P_i; these places are obtained by reversing the mapping used earlier to derive the directed graph from the PNPM. Consider our initial example of the customer-processing system shown in Figure 5 and its corresponding PNPM (cf. Figure 6). In this example we have only one service cycle: Server 0, Travel 0, Server 1, Travel 1, Server 2, Travel 2, Server 0.

Now, using the latter information, we wish to accurately represent the flow of customers in the system. We begin by retrieving the sets of input and output transitions of the place $p_H^{(i)}$ that corresponds to the head of the ith service cycle. For each transition t_{in} contained in $\bullet p_H^{(i)}$ we obtain its input place[6]and similarly, for each transition t_{out} contained in $p_H^{(i)}\bullet$ its output place. Here $\bullet p$ and $p\bullet$ denote the sets of input and output transitions of a place p. Similarly, $\bullet t$ and $t\bullet$ denote the sets of input and output places of a transition t. In our example (cf. Figure 6) we have: $p_H^{(i)} = \text{Server } 0$, $\bullet\text{Server } 0 = \{t_7, t_9\}$, $\text{Server } 0\bullet = \{t_1, t_4\}$, $\bullet t_7 = \{\text{Travel } 2\}$, $\bullet t_9 = \{\text{Arrived}\}$, $t_1\bullet = \{\text{Travel } 0\}$, and $t_4\bullet = \{\text{Intermediate}\}$.

The next step is to determine the transition t_{in}, $t_{in} \in \bullet p_H^{(i)}$, whose input place is in cycle i, and change its firing mode. In particular, we only need to modify its forward incidence function to support another token type. That is,

$$I^+(p_H^{(i)}, t_{in})(black) = 1 \rightarrow I^+(p_H^{(i)}, t_{in})(c) = 1$$

for some (hitherto unused) colour c other than black. We then identify the transition t_{out}, $t_{out} \in p_H^{(i)}\bullet$, whose output place p_k is not in cycle i, and change both its backward and forward incidence functions accordingly, i.e.

$$I^-(p_H^{(i)}, t_{out})(black) = 1 \rightarrow I^-(p_H^{(i)}, t_{out})(c) = 1$$

$$I^+(p_k, t_{out})(black) = 1 \rightarrow I^+(p_k, t_{out})(c) = 1.$$

[5]When two vertices $v_i, v_j \in V$ are adjacent through the edge $(v_i, v_j) \in E$, then v_j is called a successor of v_i and v_i is called a predecessor of v_j [6].

[6]At this point, each transition in the model has only one input and only one output place.

We proceed by recursively changing the backward and forward incidence functions of all subsequent connected transitions in the non-parameterised model; we consider two transitions t_i, t_j to be connected if $t_i \bullet \cap \bullet t_j \neq \emptyset$. This process continues until we reach the transition whose output place is the Repository. For this transition we only change its backward incidence function since we wish the original token type to be restored in the Repository. Table 2 lists the required changes of the transitions' firing mode for our example. The resulting model is shown in Figure 7.

3.4 Service Cycles and Multiple Customer Classes

When multiple customer classes exist and in particular, when at least two classes of customers perform the same service cycle, the modelling approach presented in Section 3.3 is not applicable. Here, we demonstrate the reason the latter approach fails and propose a solution.

Consider the customer-processing system shown in Figure 5 and assume the existence of two customer classes. Following the methodology presented in Section 3.2 we obtain the CGSPN model depicted in Figure 9. If we were to map the places and transitions of that model to vertices and edges to form the corresponding directed graph (as described in Section 3.3), we would obtain a directed multigraph, also known as p-graph with $p = 2$ (see Figure 10). In a p-graph, p denotes the maximum number of arcs having the same initial and terminal vertices. If we have a system with three customer classes where each class performs the same service cycle, the corresponding graph of the PNPM for that system would be a 3-graph. While Theorem 1 is still applicable for this graph, Johnson's algorithm considers the three elementary cycles as being the same and thus outputs only one. The original approach fails since we can neither generalise the detected service cycle to all customer classes, nor deduce the class which performs the service cycle.

To resolve this issue we obtain the directed graph that represents the flow of customers of each class – one graph for each customer class – and then iteratively apply the same approach as before, on each graph. To find these graphs we proceed as follows. Assuming that we have N customer classes in the system, we decompose the set of the model's transitions T into disjoint subsets T_{c_i}, where c_i, $i = 1, \ldots, N$, denotes the token type that a transition's firing mode supports – this is unique since at this stage of model construction there is a one-to-one correspondence between each customer class and each token type. Of course when the modelling approach presented in Section 3.3 is applied, a one-to-many, in particular $1 : k + 1$, correspondence will exist between each customer class and each token type. Here k denotes the number of service cycles performed by a particular customer class. Then, for each T_{c_i} – along with the connected set of places – we apply the same mapping as before to derive the directed graph that corresponds to the jour-

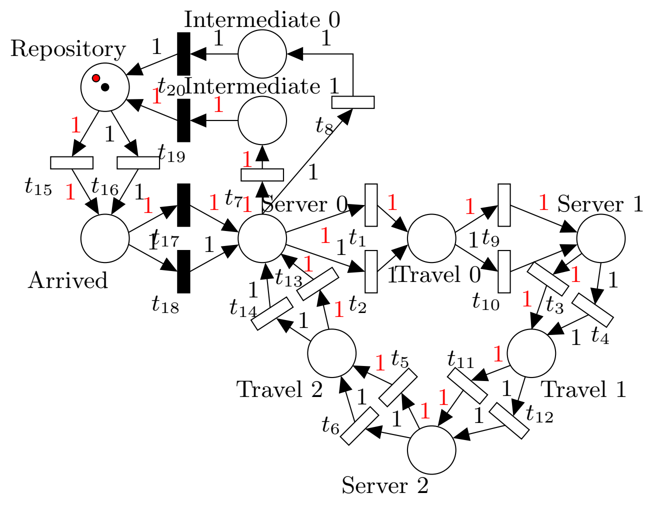


Figure 9: The CGSPN model (non-parameterised version) for the system depicted in Figure 5 when two customer classes exist. Its corresponding 2-graph is shown in Figure 10.

ney undertaken by each customer class. In our example, we assumed two classes of customers represented by black and red tokens respectively, i.e. $N = 2$, $c_1 :=$ black, $c_2 :=$ red. If we denote the set of edges that were formed from the mapping of T_{black} as E_{black}, and similarly those formed from the mapping of T_{red} as E_{red}, then the directed graphs that correspond to the first and second customer classes are, respectively, $G_1 = (V, E_{\text{black}})$ and $G_2 = (V, E_{\text{red}})$ (see Figure 10).

Figure 11 shows the CGSPN model obtained via this extended version of our initial approach for the latter example.

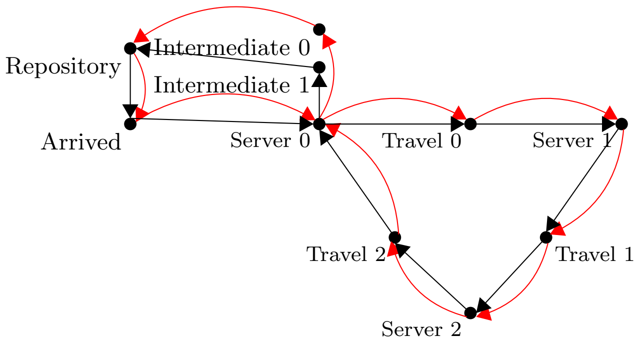


Figure 10: The directed 2-graph produced for the CGSPN model depicted in Figure 9. If E_{black} and E_{red} denote the set of black and red edges then the 2-graph can be defined as $G = (V, E_{\text{black}} \cup E_{\text{red}})$. The directed graphs for the first and second customer classes are $G_1 = (V, E_{\text{black}})$ and $G_2 = (V, E_{\text{red}})$ respectively.

3.5 Calculation and Representation of Inter-routing Probabilities

This section presents the calculation and representation of inter-routing probabilities of the customer flow within the system, when multiple classes of customers exist.

During the response and travelling time extraction process (third stage of the pipeline) we count the total number of customers of each class who received service at each service area. For each service area S_i we maintain tables – one for each customer class – whose tuples consist of two fields: the identity of another service area S_k, $k \neq i$, and the number of customers who visited S_k immediately after they completed their service at S_i. To calculate the routing probability from S_i to S_k for some k, $k \neq i$, we divide the number of customers of class j who visited S_k immediately after S_i, say $n_{i,k}^{(j)}$, by the total number of customers of the same class who were serviced by S_i, $n_i^{(j)}$. The routing of each customer class j is represented in the model as follows:

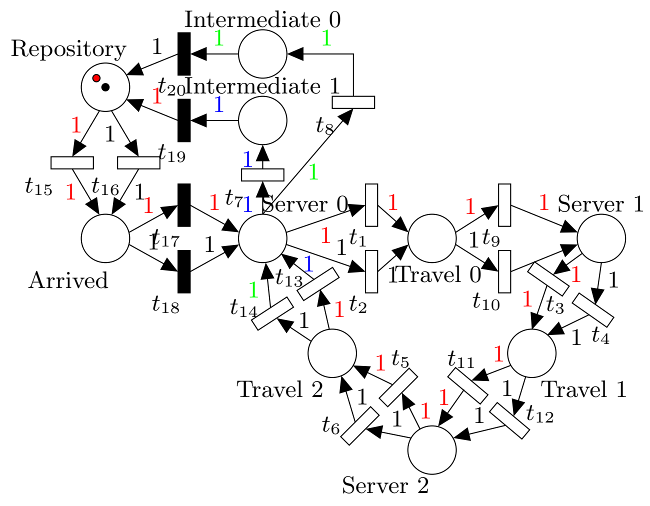


Figure 11: The CGSPN model (non-parameterised version) for system depicted in Figure 5 when two customer classes exist. Black and green tokens are used to distinguish between unserviced and serviced customers of class one. Red and blue tokens are used to distinguish between unserviced and serviced customers of class two.

1. For each service area S_i and for each customer class j, an intermediate place is created, where tokens are placed after the associated service time transition fires.
2. For each destination service area S_k of each origin service area S_i, immediate transitions are created from the intermediate places of S_i to a so-called "travel" place. Their weights are set equal to $n_{i,k}^{(j)}/n_i^{(j)}$.
3. Travelling time transitions are created, connecting each constructed travel place to the corresponding destination service area S_k.

4. CASE STUDY

We present a case study to evaluate the pipeline's newly added features and their integration in the existing methodology. For this case study we have generated location tracking data using an extended version of LocTrackJINQS [9]. Synthetic data provides two advantages over real traces: it allows us to characterise the degree of accuracy of the inferred distributions and their parameters – as the exact model parameters and processes are known – and its generation is performed in a time efficient manner rather than engaging in long experimental procedures.

Here, a simplified model of an A&E department is employed. It comprises five service areas: a reception (S1), an examination room (S2), an x-ray operation room (S3), an x-ray room (S4) and a treatment room (S5). The simulation takes place in a 40 m × 28 m virtual environment. Figure 12 depicts the experimental setup. Each service area consists

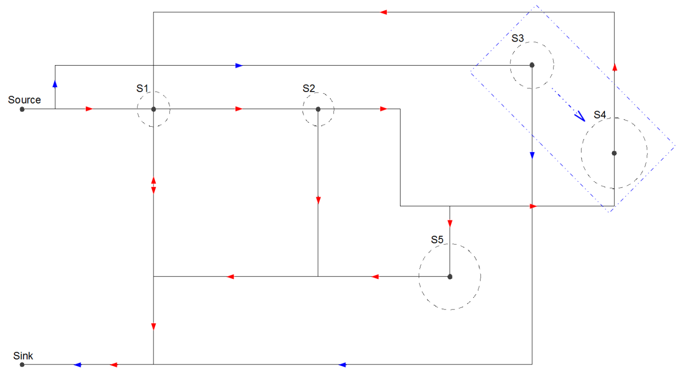


Figure 12: The experimental setup in terms of abstract system structure for this case study. The red arrows represent the flow of customer classes 0 and 1. Blue arrows are used to indicate the flow of customer classes 2 and 3. The blue dotted arrow (contained in the blue dotted rectangle) indicates the synchronisation conditions imposed by S3 on S4: one nurse (customer class 2) and one radiologist (customer class 3) must be present in the x-ray operation room (S3) so that the screening process of patients in the x-ray room (S4) can be initiated. After service completion at S4 and S5, customers of classes 0 and 1 obtain service from S1 (for the second time) and then exit the system, thus defining two service cycles.

of a single customer-processing server with a random service discipline. The location update error is normally distributed with mean 0.15 m and standard deviation 0.2 m.

We assume minor and major patients classes which correspond to customer classes zero and one respectively. Furthermore, two more classes of customers exist: nurses (customer class two) and radiologists (customer class three). The arrival process for each customer class is shown in Table 3. Patients entering the system are routed to the reception in order to register and then proceed to the examination room. From there they get discharged, or are sent to the x-ray room (S4) or to the treatment room (S5). This routing occurs with probability 0.3, 0.4 and 0.3 respectively. Patients routed to the x-ray or treatment room must then revisit the reception to schedule a followup examination before exiting the system, i.e. we have two service cycles: S1, S2, S4, S1, and S1, S2, S5, S1. Nurses and radiologists are immediately routed to the x-ray operation room (S3) and when their service is terminated they exit the system. In order for the scanning process of patients to initiate and progress, at least one nurse and one radiologist must be present in S3. The customers are assumed to travel between service areas with speed drawn from a normal distribution whose parameters – for each class – are shown in Table 3. The service time for each server and each customer class may follow a different density function. Table 4 shows each service area's actual location and service radius, as well as its service time density for each customer class.

Customer Class	Arrival Distribution	Speed Distribution
0	Exp(0.01)	Normal(0.38, 0.1)
1	Exp(0.008)	Normal(0.25, 0.1)
2	Exp(0.004)	Normal(0.4, 0.2)
3	Exp(0.004)	Normal(0.5, 0.15)

Table 3: The customer arrival process and speed distribution for each customer class.

	Server Location	Service Radius	Service Time Density	
S1	(10.0, 12.5)	1.0	Customer Class 0	Exp(0.033)
			Customer Class 1	Exp(0.05)
S2	(20.0,12.5)	0.95	Customer Class 0	Erlang(2, 0.035)
			Customer Class 1	Normal(40, 10)
S3	(33.0,10.0)	1.33	Customer Class 2	Erlang(6, 0.05)
			Customer Class 3	Erlang(5, 0.04)
S4	(38.0,15.0)	1.8	Customer Class 0	Exp(0.033)
			Customer Class 1	Exp(0.033)
S5	(28.0,22.0)	1.5	Customer Class 0	Exp(0.013)
			Customer Class 1	Erlang(2, 0.013)

Table 4: The parameters for each service area in the system, for each customer class.

4.1 Results

The locations and service radii of the service areas are in general approximated well (see Table 6). The largest error for the service area location inference – specified in terms of the distance between the real and inferred points – is 0.320 metres. The maximum service radius approximation error is 0.277 metres.

Service Area	Customer Class	Simulated Probability	Inferred Probability (3 d.p.)
S4	0	0.4	0.429
	1	0.4	0.397
S5	0	0.3	0.280
	1	0.3	0.298
Sink	0	0.3	0.291
	1	0.3	0.305

Table 5: The simulated and inferred routing probability (to three decimal places) of the customer flow from service area S2.

The quality of the extracted service time samples and the best-fit HErD are assessed by Kolmogorov-Smirnov tests which examine their compatibility (see Table 7). To examine the degree of accuracy of the approximation of the theoretical (simulated) service time distribution by the best-fit HErD we compute the relative entropy [13] of the two distributions. As shown in Table 8, good level of agreement is observed between the parameters of the best-fit HErD and the parameters of the corresponding theoretical service time distribution (relative entropy for all fitted HErDs is bounded by 0.119 nat). Figure 14 shows the cumulative histogram of the extracted service time samples and its best-fit HErD compared with the corresponding theoretical distribution for customer class 0.

The inferred CGSPN performance model is shown in Figure 13. Customer classes 0, 1, 2 and 3 are represented by

	Server Location			Service Radius		
	Real	Inferred	Error	Real	Inferred	Error
S1	$(10.0, 12.5)$	$(9.955, 12.511)$	0.046	1.0	1.095	0.095
S2	$(20.0, 12.5)$	$(20.014, 12.546)$	0.048	0.95	1.025	0.075
S3	$(33.0, 10.0)$	$(33.017, 10.014)$	0.022	1.33	1.454	0.124
S4	$(38.0, 15.0)$	$(37.814, 14.976)$	0.188	1.8	2.077	0.277
S5	$(28.0, 22.0)$	$(28.136, 22.290)$	0.320	1.5	1.609	0.109

Table 6: The inferred location and service radius for each service area in the system accompanied with the absolute error. These values are given to three decimal places.

red, blue, green and black coloured tokens. Places labelled as Server 0, Server 1, Server 3, Server 2 and Server 4 correspond to service areas S1, S2, S3, S4 and S5. The two service cycles involving service areas S1, S2, S4 and S1, S2, S5, have been correctly identified and modelled. In particular, the CGSPN subnets with labels HErD21 and HErD15 connect travel places, associated with customer movement after S4 and S5, to S1 thus completing the service cycles for customer class 0 (subnets HErD19 and HErD13 for customer class 1). We note the token colour change for the first customer class (class 0) from red to brown (by HErD21 and HErD15) and for the second customer class (class 1) from blue to yellow (by HErD19 and HErD13). Table 5 shows the inferred routing probability from S2 to S4, S5 and Sink (represented by the Repository place). Also, the synchronisation conditions between S3 (synchronising service area) and S4 (synchronised service area) have been modelled correctly; one nurse (green colour) and one radiologist (black colour) are required to be present in S3 so that patients in S4 can be scanned.

5. CONCLUSION AND SUMMARY

This paper has introduced extensions to an existing data processing pipeline for the automated derivation of performance models from location tracking traces. These increase its domain of applicability to customer-processing systems supporting multiple customer classes and service cycles. The calculation and representation of inter-routing probabilities of the customer flow between the system's service areas, when multiple customer classes exist, has also been presented. All extensions are incorporated within the third stage of the pipeline.

The case study results indicate that the extended methodology can infer the stochastic features of certain multi-class systems accurately, even in the presence of synchronisation and service cycles, at least when synthetically-generated location tracking data is used.

Our current work assumes single-server semantics, random service discipline and static customer classes. In future work we intend to relax these assumptions. It should be straightforward to modify the subnets used to model the service times of service areas (cf. Figure 3) to support multiple servers by changing the number of tokens contained in the complementary place which controls the number of tokens (customers) which can be simultaneously in the subnet[7]. More challenging tasks include devising a means to infer the number of servers present in each service area from location tracking data, and supporting dynamic customer classes (where class changes may occur after completion of service at a service area).

		Class 0		Class 1	
S1 - 1st visit	Test Statistic	0.0741		0.0638	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.0807	0.0920	0.0877	0.0999
	Compatible ?	Yes	Yes	Yes	Yes
S1 - 2nd visit	Test Statistic	0.0798		0.0422	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.0973	0.1109	0.1110	0.1265
	Compatible ?	Yes	Yes	Yes	Yes
S2	Test Statistic	0.0479		0.0455	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.0809	0.0922	0.0901	0.1027
	Compatible ?	Yes	Yes	Yes	Yes
S4	Test Statistic	0.1095		0.1293	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.1244	0.1418	0.1443	0.1645
	Compatible ?	Yes	Yes	Yes	Yes
S5	Test Statistic	0.0928		0.1011	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.1529	0.1743	0.1671	0.1905
	Compatible ?	Yes	Yes	Yes	Yes
		Class 2		Class 3	
S3	Test Statistic	0.0708		0.1374	
	α	0.1	0.05	0.1	0.05
	Critical Values	0.1370	0.1562	0.1430	0.1630
	Compatible ?	Yes	Yes	Yes	Yes

Table 7: Kolmogorov-Smirnov test at significance levels 0.1 and 0.05 applied to the extracted service time samples for each service area in this case study. The null hypothesis is that each extracted sample belongs to the corresponding best-fitted HErD.

Ultimately, we would like to examine the applicability and performance of our pipeline in the context of more complicated scenarios using location tracking data gathered by a real RTLS. In fact, experiments on a real A&E department would be of particular interest given the favourable outcome of the synthetic A&E case study presented in this paper.

6. REFERENCES

[1] H. Akaike. A New Look at the Statistical Model Identification. *Automatic Control, IEEE Transactions on*, 19(6):716 – 723, December 1974.

[2] N. Anastasiou, T.-C. Horng, and W. Knottenbelt. Deriving Generalised Stochastic Petri Net performance models from High-Precision Location Tracking Data. In *Proc. 5th Intl. Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2011)*, Paris, France, May 2011.

[3] N. Anastasiou, W. Knottenbelt, and A. Marin. Automatic Synchronisation Detection in Petri Net Performance Models Derived from Location Tracking Data. In *Proc. 8th European Performance Engineering Workshop*, pages 29–41, 2011.

[4] F. Bause and P. Kritzinger. *Stochastic Petri Nets.* Friedrich Vieweg & Sohn Verlag, 2002.

[7] For timed transitions contained in these subnets we assume infinite-server semantics.

	Service Time Density		Fitted HErD Parameters			Relative Entropy
			Phase Lengths	Rate (3 d.p.)	Weights (3 d.p.)	(3 d.p.)
S1 - 1st visit	Class 0	Exp(0.033)	1	0.034	1.000	0.001
	Class 1	Exp(0.05)	2, 2	0.084, 1.184	0.899, 0.101	0.054
S1 - 2nd visit	Class 0	Exp(0.033)	1	0.034	1.000	0.001
	Class 1	Exp(0.05)	1, 2	0.070, 0.088	0.325, 0.675	0.045
S2	Class 0	Erlang(2, 0.035)	2, 8	0.032, 5.686	0.985, 0.015	0.017
	Class 1	Normal(40, 10)	17	0.385	1.000	0.119
S3	Class 2	Erlang(6, 0.05)	5	0.037	1.000	0.049
	Class 3	Erlang(5, 0.04)	5	0.036	1.000	0.030
S4	Class 0	Exp(0.033)	1	0.027	1.000	0.021
	Class 1	Exp(0.033)	1	0.027	1.000	0.016
S5	Class 0	Exp(0.013)	1	0.013	1.000	0.000
	Class 1	Erlang(2, 0.013)	2	0.013	1.000	0.001

Table 8: The HErD parameters fitted by G-FIT for each server's service time density and each customer class, along with the relative entropy (in nat) between the theoretical and fitted probability density function. The rates and weights of each fitted HErD, as well as the relative entropy values are given to three decimal places.

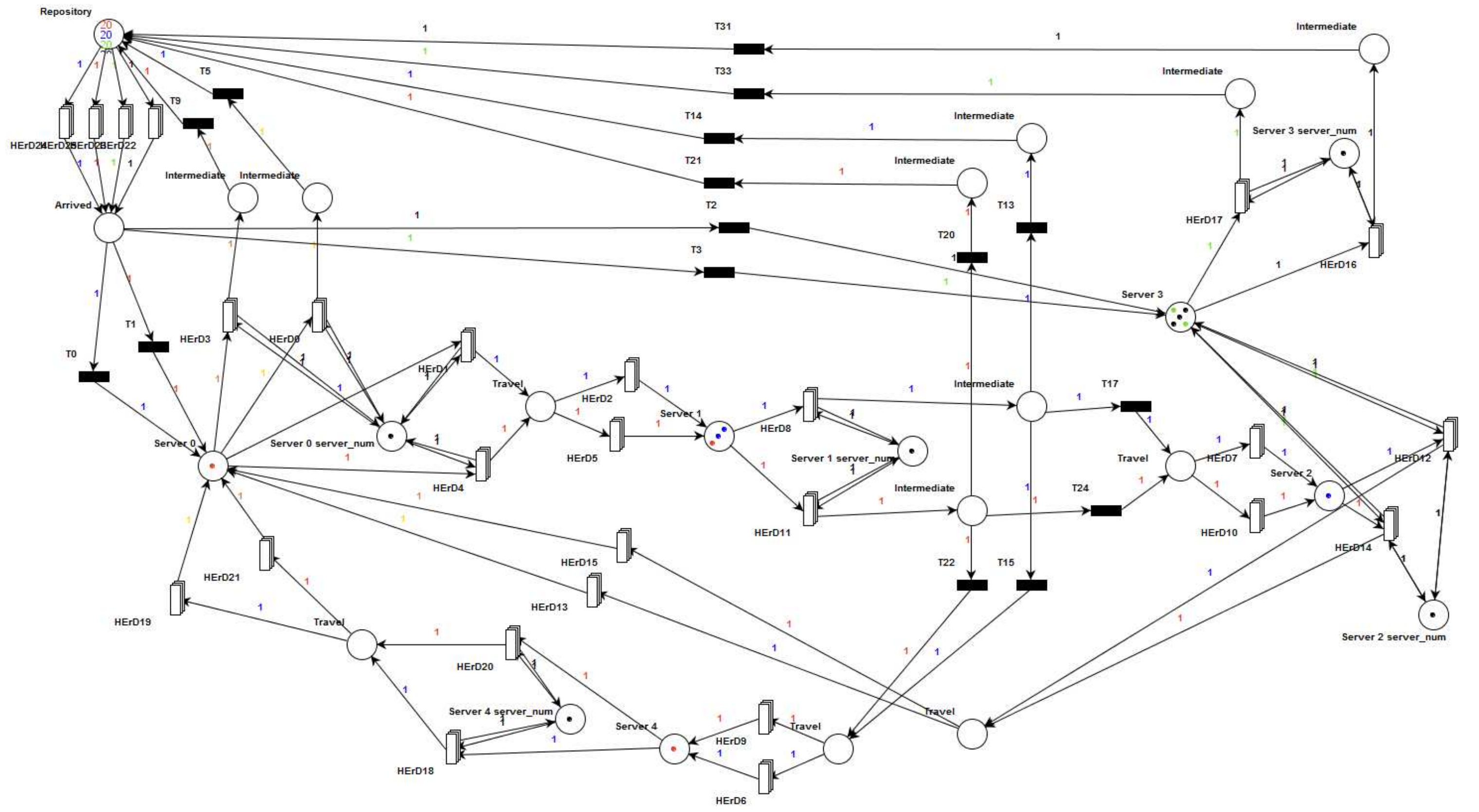


Figure 13: The inferred CGSPN performance model for this case study, visualised in PIPE2, (in compact transition form). Red, blue, green and black tokens represent customer class 0, 1, 2 and 3 in respective order.

[5] P. Bonet, C. Llado, R. Puijaner, and W. Knottenbelt. PIPE v2.5: A Petri Net Tool for Performance Modelling. In *Proc. 23rd Latin American Conference on Informatics (CLEI 2007)*, San Jose, Costa Rica, October 2007.

[6] B. Carré. *Graphs and networks*. Clarendon Press, 1979.

[7] N. Dingle, W. Knottenbelt, and T. Suto. PIPE2: A tool for the performance evaluation of Generalised Stochastic Petri Nets. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):34–39, 2009.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, 1996.

[9] T.-C. Horng, N. Anastasiou, and W. Knottenbelt. LocTrackJINQS: An Extensible Location-aware Simulation Tool for Multiclass Queueing Networks. In *Proc. 5th Intl. Workshop on Practical Applications of*

Stochastic Modelling (PASM 2011), Karlsruhe, Germany, March 2011.

[10] T.-C. Horng, N. Dingle, A. Jackson, and W. Knottenbelt. Towards the Automated Inference of Queueing Network Models from High-Precision Location Tracking Data. In *Proc. 23rd European Conference on Modelling and Simulation (ECMS 2009)*, pages 664–674, May 2009.

[11] D. Johnson. Finding All the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

[12] S. Kounev et al. Automated Simulation-Based Capacity Planning for Enterprise Data Fabrics. In *Proc. 4th Intl. Conference on Simulation Tools and Techniques (SIMULTOOLS 2011)*, pages 27–36, 2011.

[13] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[14] M. Li and N. Georganas. Colored Generalized Stochastic Petri Nets for Integrated System Protocol Performance Modelling. In *Proc. GLOBECOM 1988*, volume 3, pages 1798–1802, 1988.

[15] M. Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.

[16] A. Thümmler, P. Buchholz, and M. Telek. A Novel Approach for Phase-Type Fitting with the EM Algorithm. *IEEE Transactions on Dependable and Secure Computing*, 3:245–258, 2005.

[17] Y. Xue, R. Kieckhafer, and F. Choobineh. Automated construction of GSPN models for flexible manufacturing systems. *Computers in Industry*, 37:17–25, 1998.

APPENDIX

Proof of Theorem 1.

PROOF. We consider a physical customer-processing system consisting of the set of service areas $S = \{S_i \mid i = 1, \dots, N\}$, for some finite $N \in \mathbb{N}$. Assume that a service cycle exists, and consider the directed graph which corresponds to the PNPM (non-parameterised version) of the underlying system.

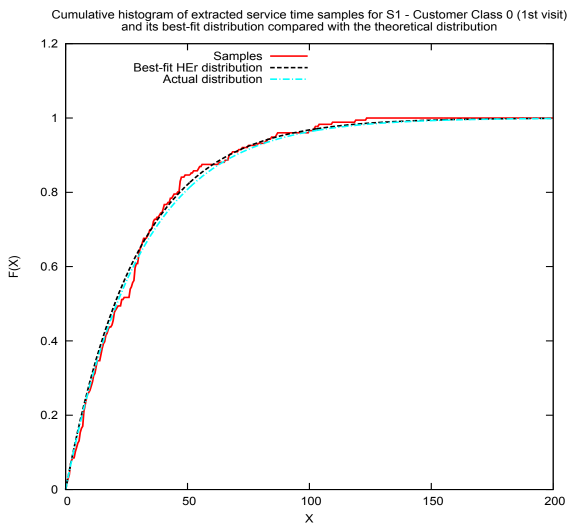
We define:

1. $S_c \subseteq S$ to be the subset of service areas that make up the service cycle,
2. $P_c = \{v_j, v_{j+1}, \dots, v_{j+k}\}$ to be the set of vertices which correspond either to the places representing service areas $S_i \in S_c$ or to the travel places that exist between pairs of service areas S_i, S_k, $i \neq k$, $S_i, S_k \in S_c$.

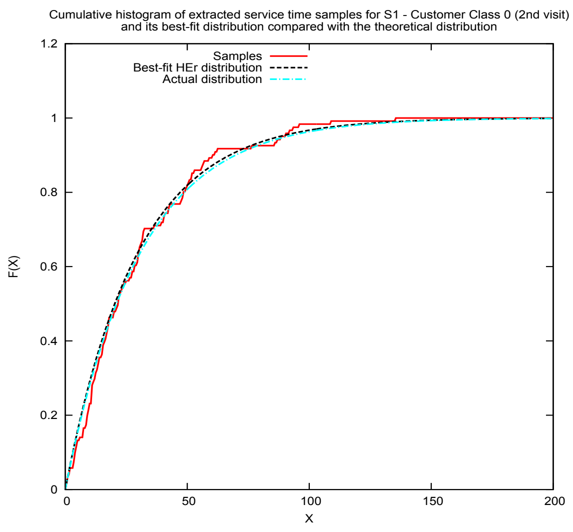
We examine the two conditions of the theorem individually:

At least two predecessors are required.
If no predecessor of the vertex v_j which corresponds to S_j, exists, then it is trivial that no service cycle exists. Now let us assume that only one predecessor of the vertex v_j exists, say v_p with $p \neq j$. We need to consider two cases. If $v_p \in P_c$, it corresponds to the travel place which connects the last service area of the service cycle, say S_l, $l \neq j$, $S_l \in S_c$, to the head of the service cycle S_j. This means that the customers initiated their service sequence at the service area $S_l \in S_c$. But since S_j is the head of the service cycle and $j \neq l$, we reach a contradiction. If $v_p \notin P_c$, then v_j is not accessible from any vertex $v \in P_c$, and consequently customers cannot reach S_j from the last service area of the service cycle, meaning that no service cycle exists. Thus, we obtain a contradiction.

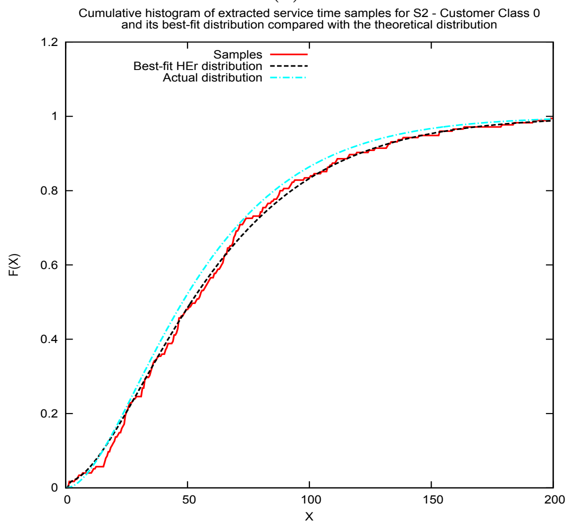
At least two successors are required.
If no successor of the vertex v_j which corresponds to S_j exists, then it is trivial that no service cycle exists. Now assume that only one successor of the vertex v_j exists, say v_p, $p \neq j$. If $v_p \in P_c$, it corresponds to the travel place which connects S_j to the next service area in the service cycle, say S_{j+1}. Since no other vertex is a successor of v_j this means that all customers completing their service cycle will be rerouted to S_{j+1}, thus repeating the same service cycle. This violates the second condition of definition 1 and therefore, a contradiction is obtained. If $v_p \notin P_c$, then none of the vertices corresponding to service areas S_l, $l = j+1, \dots, j+k$, are accessible from S_j, hence no service cycle exists. Again, we reach a contradiction. $\Box$



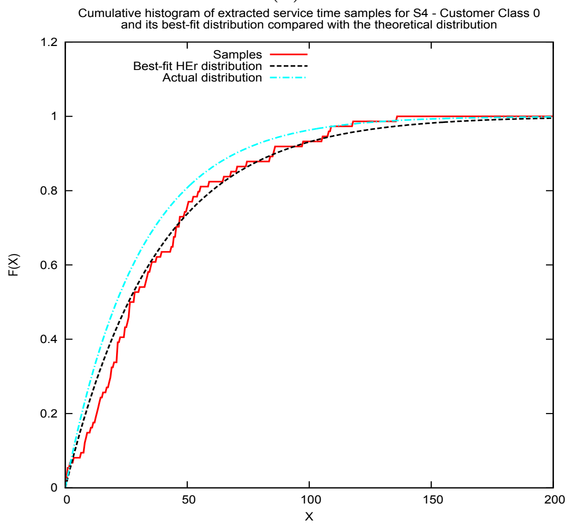
(a)



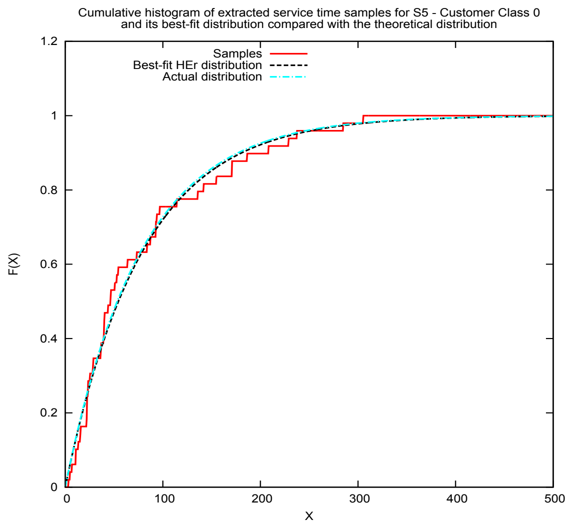
(b)



(c)



(d)



(e)

Figure 14: Case Study: Graphs 14(a), 14(b), 14(c), 14(d) and 14(e) show the cumulative histogram of the extracted service time samples (adjusted for synchronisation in 14(d)) for customer class 0 and its best-fit hyper-Erlang distribution compared with the corresponding theoretical distribution for S1 (entry to service cycle), S1 (exit from service cycle), S2, S4 and S5 respectively.