Mining Resource Scheduling Protocols

Arik Senderovich[1], Matthias Weidlich[2], Avigdor Gal[1], and Avishai Mandelbaum[1]

[1] Technion – Israel Institute of Technology
{sariks@tx,avigal@ie,avim@ie}.technion.ac.il
[2] Imperial College London
m.weidlich@imperial.ac.uk

Abstract. In service processes, as found in the telecommunications, financial, or healthcare sector, customers compete for the scarce capacity of service providers. For such processes, performance analysis is important and it often targets the time that customers are delayed prior to service. However, this wait time cannot be fully explained by the load imposed on service providers. Indeed, it also depends on resource scheduling protocols, which determine the order of activities that a service provider decides to follow when serving customers. This work focuses on automatically learning resource decisions from events. We hypothesize that queueing information serves as an essential element in mining such protocols and hence, we utilize the queueing perspective of customers in the mining process. We propose two types of mining techniques: advanced classification methods from data mining that include queueing information in their explanatory features and heuristics that originate in queueing theory. Empirical evaluation shows that incorporating the queueing perspective into mining of scheduling protocols improves predictive power.

1 Introduction

Service processes can be viewed as a special case of business processes, in which service consumers (aka customers) compete for the scarce capacity of service providers [1]. Service processes can be found, for instance, in the telecommunications, financial, or medical sectors. The handling of customers by a call center or the treatment of patients in an emergency department at a hospital are examples of a service process.

Service processes often face high volumes of service requests, which are subject to large variations over time and high level of uncertainty in the amount of incoming demand [2]. Therefore, in order to assure successful operation of a service process (e.g. limit the time that customers wait for service), one must set an adequate level of resource capacity. Operational analysis, that is based on Queueing Theory [3, 4], is common practice for setting capacity levels that would accommodate the time-varying and random demand. Given that service processes are often supported by information systems, event logs recorded during process execution can be exploited for such performance-driven analyses. Various techniques for operational process mining addressed the prediction of wait times based on characteristics of service requests [5, 6]. Recently, it was argued that the load imposed on service providers is essential for performance prediction. To account for delays in processing that stem from queueing of customers, queueing models and related predictors can be constructed from event data [7].

In this work, we start with the observation that delays in processing cannot be fully explained by the process load. Rather, delays also originate from service providers. In fact, service providers play a symmetric role in their influence on process performance, when compared to customers. Service providers follow *resource scheduling protocols* that define the order of activities followed when serving customers. Knowledge of these protocols allows for predicting the next task of a service provider, out of a set of feasible tasks, thereby providing a more complete performance analysis and improving the accuracy of wait time prediction.

Following the theme of operational process mining, this paper sets out to mine scheduling protocols of service providers from recorded event data. We hypothesize that queueing information serves as an essential element in mining such protocols and hence, one must utilize the queueing perspective of customers in the mining process. We consider two approaches for mining of resource scheduling protocols. First, we show how data mining methods can be employed when queueing information is considered as part of the explanatory features. Second, we present heuristics that originate in queueing theory and do not require historical data in their application. Both techniques are evaluated empirically using real-world logs of a large Israeli telecommunication company. The data covers three months of operation of a service process, with up to 50,000 requests per day. Our results indicate that queueing heuristics, as well as decision trees and random forests, are superior to other methods. Since queueing heuristics are intuitive and can be facilitated online (without a preliminary learning phase), we argue that these are less time-and-resource consuming than data mining methods.

To conclude, this paper makes the following contributions:

- It puts forward the duality of service consumers and service providers, to reach to a more holistic analysis of the performance of service processes. Resource scheduling protocols at the side of the provider are identified as a major influencing factor.
- It provides a set of mining techniques to extract resource scheduling protocols from event data, recorded during process execution. Those techniques vary in the degree with which they incorporate the queueing perspective of customers.
- It reports on a comprehensive evaluation of the proposed techniques using real-world logs. In particular, we show that simple techniques that include basic queueing information are competitive with respect to advanced data mining techniques that require an offline learning phase.

The remainder of the paper is structured as follows. The next section provides further background on service processes and possible causes for delays. Section 3 introduces our model. Section 4 presents algorithms for mining various types of protocols from event data. We evaluated our approach with real-world data in Section 5. Section 6 reviews our contributions in the light of related work, before we conclude in Section 7.

2 Background

Service Processes as Interacting Processes. Service processes show some inherent duality in the sense that both service consumers and providers execute certain activities to reach a goal. Hence, it is natural to view service processes as a choreography [8], i.e., two interacting processes. An example of this view is given by the BPMN [9] model

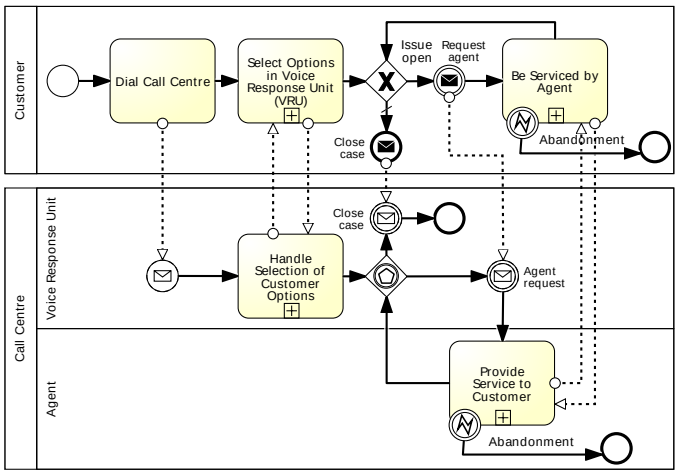

Fig. 1: A call center process modeled as interacting consumer and provider processes.

in Fig. 1. It shows a simple call center process in which a customer dials in and first interacts with a Voice Response Unit (VRU). If customers cannot solve their inquiry using the VRU, they are referred to a call center agent. Then, to handle the customer inquiry, multiple iterations with agents may be required.

The model in Fig. 1 highlights the interaction of service consumer and provider processes. However, it focuses only on the activities executed by the service provider that are specifically required to handle a single request of a service consumer. The model neglects the fact that resources at the side of the service provider, i.e., the agent in our example, also follow an explicit process when handling requests of service consumers. This perspective of our example is highlighted in Fig. 2, which depicts how an agent chooses a customer who waits for being served, serves the customer, and then repeats the whole procedure until the shift is finished.

Delays in Service Processes. The various illustrated perspectives on a service process are important when assessing its performance in general, and investigating the causes of delays in processing in particular. Since service consumers typically compete for the scarce capacity of service providers, clearly, unavailability of resources at the service provider is a first cause of delays. In this case, customers are queued and served later when resources become available. The process of the agent in Fig. 2, however, also highlights that there is a particular activity that refers to the choice of the customer to be served. Consequently, the selection strategy applied by an agent is a second cause for delays in processing. Even in case agents are available to handle customer requests, a particular request may be delayed because of this selection strategy.

Resource Scheduling Protocols. Strategies for the selection of resources, commonly referred as *scheduling protocols*, can be grounded on various aspects. Examples include the properties that relate to the processing state of the service provider (e.g., how long a provider was busy or idle), attributes of the process instances (e.g., scheduling based on types of customers and service providers), or global context (e.g., scheduling that differs on working days and holidays).

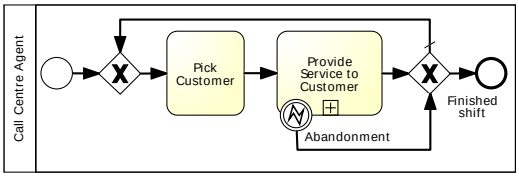

Fig. 2: Process from the perspective of an agent.

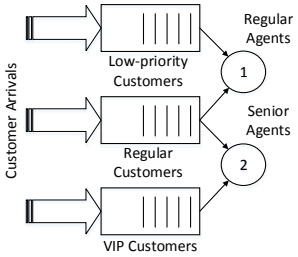

Fig. 3: Queueing perspective for types of customers and agents.

For our example, the call center service process, scheduling based on attributes of process instances is illustrated in Fig. 3. Here, customers are assumed to fall in one out of three groups, 'low-priority', 'regular', or 'VIP'. Customers of either type end up in a separate queue. Agents, in turn, are classified based on their experience, so that agents are 'regular' or 'senior'. In our example, regular agents may serve low-priority or regular customers, whereas senior agents serve regular or VIP customers. This creates a queueing system that is referred to have a W-architecture; see [10]. For this setting, different scheduling protocols may be implemented. As an example, senior agents could always prefer VIP customers and select regular customers only if the VIP queue is empty. A different protocol could define that senior agents prefer VIP customers, but also select the first customer in the regular queue if they waited for more than 15 minutes, and the VIP queue contains at most 2 customers that waited for less than 2 minutes.

These examples illustrate that deriving resource scheduling protocols from process execution data is valuable to achieve more holistic performance analysis. In this work, we hypothesize that queueing information is particularly useful in extracting such protocols.

3 Service Logs and the Protocol Mining Problem

In order to learn allocation of resources to customers from event data, Section 3.1 first defines our model of the *service log* (S-Log). A service log is an event log (c.f. [11, Ch. 4]) that consists of service events and paths that are related to either the customers or the resources of a service process. We then define the problem of mining resource scheduling protocols in Section 3.2. We also provide a brief overview of how to assess the quality of mined protocols in terms of their prediction error.

3.1 Service Logs

For service logs (S-Logs), we present two instantiations that reflect the duality of service processes. The first log contains events that come from the customer's perspective and include, e.g., the entry of a queue, the start of service, and abandonment. The second log consists of resource related events, e.g., the start of service, initiation of back-office work, and the start of a work break. Clearly, both types of event data are required for mining resource protocols: customer log provides us with queueing information, while the resource log presents the decisions about resource allocations.

As a first step to define an S-Log, we make the following assumptions:

- Service entities (e.g. customers, resources) go through *service paths* that consist of *service events*.
- Service events and paths must have unique identifiers (events and paths cannot have the same identifier).
- Service events have attributes.

Below, we first define the essential concepts of service logs, i.e., service events (or events for short) and service paths (or paths), and relate events to their attributes.

Definition 1 (Service event, Service path) *Denote by $\mathcal{S}$ the set of all possible service events, i.e. unique event identifiers. Let $\mathcal{S}^*$ be the set of all finite sequences over $\mathcal{S}$. We define $\Pi \subseteq \mathcal{S}^*$ as the set of all* feasible *service paths, i.e. finite sequences of service events. We require that each service event appears at most once in some path.*

According to this definition, a path $p \in \Pi$ is a finite sequence $(p_1, \ldots, p_n)$ of length n of events, such that $p_i \in \mathcal{S}, i = 1, ..., n$. It is worth to mention that, in the literature, paths are often referred to as *cases*. In this work, we wish to capture resource paths as well as customer paths, hence the extension of cases to paths.

Service events are associated with attributes, e.g., *timestamps*, *service activities*, *service locations*, and *resources*. We model such an attribute as a function that assigns an attribute value to a service event. A set of such attribute functions, in turn, defines the schema (aka structure) of a set of service events.

Definition 2 (Attribute function, Event schema) *Let A be the domain of an event attribute. Then, the attribute function $\alpha : \mathcal{S} \to A$ assigns values of this domain to service events. A finite set $\{\alpha_1, \ldots, \alpha_n\}$ of attribute functions is called an event schema.*

For example, α may be the attribute function for service activities, i.e., the domain of the function is defined as $\{$service start, service end$\}$. A specific service event $s \in \mathcal{S}$ then either indicates the start ($\alpha(s) =$ service start) or end of service ($\alpha(s) =$ service end).

Using the introduced concepts, we define the general notion of an S-Log.

Definition 3 (S-Log) *A service log (S-Log) is a tuple $(S, G, \alpha_{\mathcal{S}})$ where,*

- *$S \subseteq \mathcal{S}$ is the set of* observed *service events.*
- *$G \subseteq \Pi$ is the set of* observed *service paths.*
- *$\alpha_{\mathcal{S}}$ is the event schema.*

The notion of a service log generalizes the *functional* definition of an event log as presented in [12]. Further, it allows for capturing the duality of service processes, which is reflected in events that stem from the customer's perspective and events that relate to the resources at the service provider. Below, we define service logs of customers and resources as they are recorded in service processes that show a W-architecture, as our example in Figure 3. Either type of log follows the generic structure of a service log, but comes with different event schemas, i.e., different sets of attribute functions.

Customer Log. Recording the behavior of a customer includes information on the time at which an event was observed, the type of the customer, and the type of the executed activity. The latter may refer to the start of queueing, abandonment, the start of service, or the end of service.

Definition 4 (Customer S-Log) *A customer service log is a service log* $(S, G, \{\tau, \eta, \epsilon\})$*, where*

- $\tau : \mathcal{S} \rightarrow \mathbb{N}^+$ *is a timestamp attribute function.*
- $\eta : \mathcal{S} \rightarrow \mathbb{N}^+ \cup \{\perp\}$ *is a customer type attribute function, with* $\perp$ *being the null value.*
- $\epsilon : \mathcal{S} \rightarrow E = \{$*qEntry, qAbandon, sStart, sEnd*$\}$ *is an activity attribute function.*

Note that a customer service log does not contain information on the *identity* of particular customers since this information is not expected to be relevant for the operational analysis of service processes. Instead, only the activities performed by a particular type of customer at a certain point in time will be used for extracting queueing information.

Resource Log. The recording of the behavior of a resource captures different information compared to a customer. In addition to the time at which an event was observed, it includes information on the type of a customer served, the type of the resource (i.e., a skill group), the state of a resource, and the start and end of a particular state. In the remainder, we consider four specific resource states: (1) serving a customer, (2) ready to serve a customer (online, waiting for a customer to arrive), (3) performing offline back-office work and (4) idle (e.g. on a break). For each of these states, an event may signal the start or end of the respective state.

Definition 5 (Resource S-Log) *A resource service log is a service log* $(S, G, \{\tau, \eta, \sigma, \phi, \delta\})$*, where*

- $\tau : \mathcal{S} \rightarrow \mathbb{N}^+$ *is a timestamp attribute function.*
- $\eta : \mathcal{S} \rightarrow \mathbb{N}^+ \cup \{\perp\}$ *is a customer type attribute function.*
- $\sigma : \mathcal{S} \rightarrow \mathbb{N}^+$ *is a skill group attribute function.*
- $\phi : \mathcal{S} \rightarrow A = \{$*Serving, Ready, Back-Office, Idle*$\}$ *is a state attribute function.*
- $\delta : \mathcal{S} \rightarrow T = \{$*Start, End*$\}$ *is a state transaction attribute function.*

We observe that information on timestamps and customer types is included in both, customer and resource logs. However, the reference of a customer type in resource-related events is reasonable only if the resource is actually serving a customer. As a convention, therefore, we assume that $\eta(s) = \perp$ if $\phi(s) \neq Serving$ for all $s \in S$ of a resource log $(S, G, \{\tau, \eta, \sigma, \phi, \delta\})$.

3.2 Problem Statement

We aim at mining resource scheduling protocols. These protocols determine how to allocate resource types to queues of customers, i.e., how to select a certain customer type. Let $\mathcal{D}$ be the set of customer types and, therefore, possible resource allocations. For the W-architecture in Figure 3, these allocations would be given as $\mathcal{D} = \{$Low, Regular, VIP$\}$. Further, we write $X \in \mathbb{R}^p$ to refer to a random vector of features (explanatory variables) of dimension p, which is used as the basis of allocation. The random variable that is the real allocation of a resource is denoted by $D \in \mathcal{D}$. Using these notions, we are ready to define the problem addressed in this paper.

Problem 1 (Mining Resource-Scheduling Protocols) *The problem of mining resource scheduling protocols is to provide a protocol-function from the features vector space into*

the decision set, $\pi : \mathbb{R}^p \rightarrow \mathcal{D}$, such that π minimizes the expected prediction error with respect to some loss function $L(D, \pi(X))$.

We choose to represent the loss function as a $K \times K$ matrix $\mathbb{L}$ with K being the cardinality of $\mathcal{D}$. The matrix will be zero in the diagonal and non-negative elsewhere, with $L(k, l)$, corresponding to the cost of (mis)classifying decision $k \in \mathcal{D}$ as $l \in \mathcal{D}$. In the present first-time analysis, we use a $0 - 1$ loss function, i.e. $L(k, l) = 1$ holds if the classification is incorrect and $L(k, l) = 0$ otherwise.

The problem statement requires minimization of the expected prediction error w.r.t. the loss function. In general, the expected prediction error can be written as follows:

$$EPE = E[L(D, \pi(X))] = E_X \sum_{d \in \mathcal{D}} L(d, \pi(X)) \cdot P(D = d|X), \tag{1}$$

with the first expectation taken over the joint distribution of (D, X). That is, the expected value for the loss function is the expectation of a particular allocation given its probability under the random feature vector. Given a particular realization x of the random feature vector X, and a loss matrix of $0 - 1$, the protocol that minimizes the expected prediction error is given as,

$$\pi(x) = \arg\max_{d \in \mathcal{D}} P(D = d|X = x). \tag{2}$$

In other words, the best protocol for solving Problem 1 is the one that maximizes the posterior probability of decision D conditioned on a realization of the feature vector, namely x [13, Ch. 2.4]. As we outline in the remainder of this paper, data mining techniques can be used in an attempt to maximize this posterior probability.

4 Discovery of Resource Scheduling Protocols

The goal of resource scheduling protocols is tightly coupled with Quality-of-Service (QoS) that an organization aims to provide. For service processes, two types of QoS are of particular importance: (1) qualitative QoS, e.g. to which extent a customer receives the service that they requested, and (2) operational QoS, e.g. to which extent a customer was not delayed for too long. For illustration of the trade-off between the two types of QoS, consider again the W-architecture shown in Fig. 3. In order to increase operational QoS, regular agents are scheduled to VIP customers, thus inflicting the qualitative QoS. On the other hand, to provide better qualitative QoS, senior agents are often scheduled to serve regular customers.

Considering the above, we hypothesize that scheduling of resources, especially in service processes, is related to both the *skill* of the agent, i.e. the qualitative abilities of the resource, as well as the *system load* that has a direct influence on operational QoS. As discussed in [7], system load can be measured by queue-length or by waiting time of the most delayed customer (the head-of-line, or HOL for short). Based on these insights, our techniques for solving Problem 1 consider four levels of queueing information: (1) no queueing information; (2) lengths of queues of customers; (3) waiting time of the HOL in the customer queues; and lastly (4) a combination of the previous two levels.

Below, we first show how the two service logs, capturing a service process from the perspective of customers and resources respectively, are used to extract allocation decisions, i.e., the moment in time when a resource was allocated to a customer (Section 4.1). Each decision is associated with a vector of features including the skill of the currently allocated resource and the relevant queueing information at the moment of the allocation. Based on that information, we first apply advanced classification methods from data mining to solve Problem 1 (Section 4.2). Our second approach does not require any historical data, but relies on queueing heuristics that originate from optimal control theory for queues operating under heavy-traffic scenarios (Section 4.3). It uses information that is easily observable in service processes, such as the number of customers in each queue.

4.1 Mining Queueing Information from Service Logs

To mine allocation decisions that are associated with queueing information, we extract information on queue lengths and longest waiting time from a customer service log. Let $\boldsymbol{q}(t) = (q_{d_1}(t), \ldots, q_{d_n}(t))$ be a vector of queue lengths at time t, where $d_1, \ldots, d_n \in \mathcal{D}$ are allocation decisions (or customer types, respectively). Given a customer service log $(S, G, \{\tau, \eta, \epsilon\})$ and time t, queue length $q_{d_i}(t)$ is estimated by $\widehat{q_{d_i}(t)}$ as follows:

$$\widehat{q_{d_i}(t)} = |\{(g_1, \ldots, g_m) \in G \mid \epsilon(g_m) = qEntry \ \wedge \ \tau(g_m) \leq t \wedge \eta(g_m) = d_i\}|. \quad (3)$$

Similarly, we denote by $\boldsymbol{h}(t) = (h_{d_1}(t), \ldots, h_{d_n}(t))$ the vector of the longest waiting customers in each queue (the delays of the HOL), with $d_1, \ldots, d_n \in \mathcal{D}$ being allocation decisions. For a customer service log $(S, G, \{\tau, \eta, \epsilon\})$ and time t, this vector can estimated from the customer log in a similar manner:

$$\widehat{h_{d_i}(t)} = \min_{s \in \{s \in S \mid \epsilon(s) = qEntry \wedge \eta(s) = d_i \wedge \exists\ (g_1, \ldots, s) \in G\}} t - \tau(s), \quad (4)$$

Next, we mine allocation decisions from the resource perspective by identifying timestamps in which the resource switched their state to *Serving*. Given a resource service log $(S, G, \{\tau, \eta, \sigma, \phi, \delta\})$, the set of allocation decisions is given by

$$V = \{s \in S \mid \phi(s) = Serving \ \wedge \ \delta(s) = Start\}. \quad (5)$$

Finally, we derive a set of explanatory feature vectors, denote by $\mathcal{X}$, for the set of allocation decisions as follows:

$$\mathcal{X} = \{x = (s, \sigma(s), q_{d_1}(t), ..., q_{d_n}(t), h_{d_1}(t), ..., h_{d_n}(t)) \mid s \in V \ \wedge \ t = \tau(s)\}. \quad (6)$$

The vector elements correspond to the skill group of the allocated resource and the queueing information at the time of allocation (queue-lengths and HOL waiting times).

4.2 Data Mining Classifiers

We consider four data mining techniques that are suitable to solve Problem 1 as a classification problem. Namely, we use Linear Discriminant Analysis (LDA), Multinomial

Logistic Regression (MLR), decision trees and random forests. These methods provide us with a protocol-function π such that the decision obtained by applying π to a feature vector, $\pi(x) = d$, will have the maximal posterior probability among all other posteriors (see Section 3). Below, we briefly describe these data mining methods and then exemplify mining of resource scheduling protocols with one of the algorithms, i.e., decision trees.

Linear Classifiers: LDA and MLR. The LDA method constructs a discriminant function $\delta_d(x)$ that, given a vector of features x, selects the most probable decision, see Equation (2). The posterior probability $P(D = d|X = x)$ of the decision to select allocation d given feature vector x is rewritten following Bayes theorem:

$$P(D = d|X = x) = \frac{f_d(x) \cdot P(D = d)}{\sum_{l \in \mathcal{D}} f_l(x) \cdot P(D = l)}, \tag{7}$$

with $f_d(x) = P(X = x|D = d)$ and $P(D = d)$ being the prior of decision d. LDA assumes that for $d \in \mathcal{D}$ the density, $f_d(x)$, comes from a Gaussian distribution and that all classes $d \in \mathcal{D}$ have a common covariance matrix. From these assumptions, the discriminant function $\delta_d(x)$ of decision d is *linear* in x and depends on the prior distribution over the classes $P(D = d)$ and the parameters of the Gaussian distribution. Therefore, given a feature vector x, the LDA algorithm 'plugs' the vector into $\delta_d(x)$ for each class d and selects the decision with the highest discriminant function.

Similarly to LDA, the MLR method attempts to model a logic transformation of the posterior probabilities $P(D = d|X = x)$ by linear functions in x. However, unlike the LDA, the MLR ensures that these functions sum up to 1 and stay in the range of $[0, 1]$. Generally, the MLR requires more data observations for accuracy, while the LDA is less robust to outliers. Hence, both models can potentially be valuable (see [13, Ch 4]).

Tree-Based Classifiers: Decision Trees and Random Forests. Classification (decision) trees attempt to find m regions $R_1, \ldots, R_m$ in the feature space ($\mathbb{R}^p$) that would best explain the observed outcomes [13, Ch. 9.2]. Decision trees enjoy a low bias, yet suffer from a high variance. In order to handle the large variance of decision trees, the *random forests* algorithm, which has become popular and is considered state-of-the-art in data mining, was introduced by Breiman [14]. The idea is to grow a number of de-correlated decision trees (a forest) and to average on the result, thus reducing variance.

Exemplifying Protocol Mining with Decision Trees. To demonstrate the application and relevance of data mining methods for discovery of resource scheduling protocols, we present a data-based illustration of the method based on decision trees. To this end, we consider real-world data from an Israeli telecommunication company (further details on the data are given when presenting the comparative evaluation of the proposed methods). This company operates a service process that follows the W-architecture as discussed in Section 2 and illustrated in Fig. 3 with the aforementioned three types of customers (low-priority, regular, VIP). For this setting, Fig. 4 presents the histogram of delays for VIP customers, who had to wait prior to being served. We observe that a large proportion of customers experienced a 6 seconds delay. This is an indication of a protocol that causes VIP customers to enter service after a 6 seconds delay.

Applying the outlined approach for this data set leads to the result depicted in Fig. 5. In the graphical representation of the tree, queues (customer types or scheduling decisions, respectively) are denoted by Q_1, Q_2 and Q_3, corresponding to low-priority,

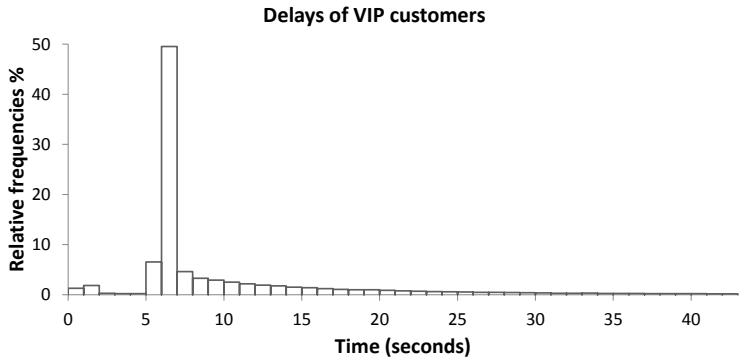

Fig. 4: Histogram of delays for VIP customers, who had to wait for service.

regular and VIP customers. The skill groups of agents are 1 and 2, i.e., regular and senior agents. HOL_Q_i represents the waiting time of the longest delayed customer in queue Q_i. Then, the protocol grounded in the decision tree reads as follows:

(1) If no information is available, the best prediction is Q_2, since this relates to the allocation of the most common customer type. In terms of posterior probabilities, we have $P(D|X) = P(D)$ (prior of D), since no feature vector is present.
(2) If the agent has skill 1 (regular agent), then the best prediction is Q_1, since the majority of customers that are allocated to this group stem from the low-priority customer queue. Otherwise, if the skill points to a senior agent, the best prediction is Q_2 based on similar considerations. Note that the first two levels of the tree do not rely on any queueing information.
(3) The second level of the tree does consider queueing information. For regular agents, the prediction is based on the fact whether there are waiting regular customers (Q_2), which should be served first. If this is not the case, prediction is based on HOL waiting times. For senior agents, the prediction is simple. They are allocated to VIP customers that wait 6 seconds or longer (the relevant node is emphasized in Fig. 5). This is exactly the phenomena that is observed in the histogram in Fig. 4.

This example illustrates how the method learns scheduling protocols from event data that are based on features of the resources and queueing information.

4.3 Queueing Heuristics

A complementary approach to the extraction of resource scheduling protocols from historical data, using data mining techniques, is the prediction of scheduling decisions based on queueing heuristics. Here we consider two such heuristics.

The first heuristic is based on the length of the queues at the time of the allocation decision. That is, given a time t, a set of feasible allocation decisions $F \subseteq \mathcal{D}$ for a resource (not every type of agent may have the skill to serve every queue), and the observed queue-length vector $\boldsymbol{q}(t) = (q_{d_1}(t), \ldots, q_{d_n}(t))$, the predicted allocation decision is defined as

$$d = \arg\max_{f \in F} q_f(t). \tag{8}$$

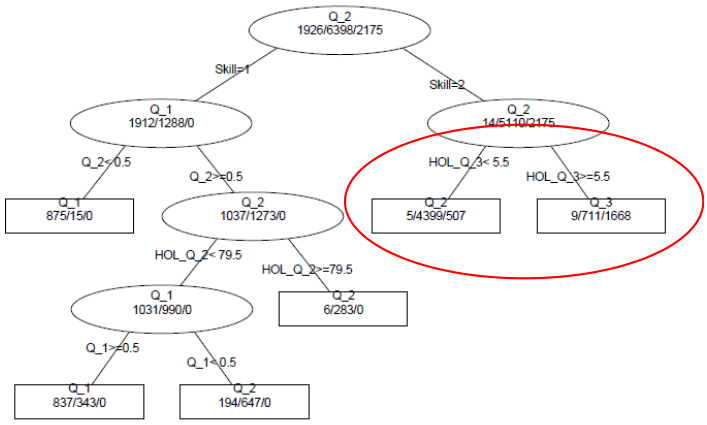

Fig. 5: Decision tree that is constructed from service logs.

We refer to this heuristic as Longest-Queue-First (LQF). This scheduling protocol has its roots in queueing control theory in heavy-traffic scenarios, and is a special case of the Fixed-Queue-Ratio rule proposed in [15]. If there are multiple queues of the same lengths (including the case in which $\boldsymbol{q}(t) = 0$ for every component), the more probable queue in terms of prior probability, is favored. For our running example and the aforementioned data set, this would allocate regular customers to senior agents and low-priority customers to regular agents.

The second heuristic follows a similar idea, but predicts the allocation to the queue with the most delayed head-of-line customer. Given a time t and the head-of-line vector $\boldsymbol{h}(t)$, the predicted allocation decision is defined as

$$d = \arg\max_{f \in F} h_f(t). \tag{9}$$

We refer to this heuristic as Most-Delayed-First (MDF). It can be shown that this heuristic is equivalent up to a constant (asymptotically in heavy-traffic) to the LQF heuristic [16]. Note that both types of heuristics ignore the priorities that exist between the three queues. We later discuss that this has to be seen as one of their limitations.

5 Evaluation

We evaluate our methods using a large-scale real-world data set. We first provide details on the data and experimental setup. Then, we report and discuss the prediction results.

Data Description. The data for our experiments stems from a call center of an Israeli telecommunication company and is gathered and stored in the Technion laboratory for Service Enterprise Engineering (SEELab)[3]. The call center processes up to 50,000 service

[3] http://ie.technion.ac.il/Labs/Serveng

requests a day, routes requests according to various resource skills, and simultaneously queues requests across multiple sites. The center is operated with around 600-800 agent positions on weekdays and 200-400 agent positions on weekends. Further, several types of services are provided; the most common are Private, Business, Technical and Content Internet. In this paper, we focus on the Private service, which handles requests with low, regular and VIP priorities. For our empirical evaluation we selected three months of data to serve as our service logs, from January 1, 2008 to March 31, 2008. The data features the events of a customer service log as well as those of a resource service log.

Experimental Setup. The controlled variable in our experiments is the *method* we apply to mine a scheduling protocol. As outlined in Section 4, our methods can be divided into: (1) data mining techniques that are based on feature vectors that may depend on queueing information and (2) queueing heuristics that rely on control theory in heavy-traffic. The uncontrolled (responding) variable in the experiments is the *misclassification rate*, i.e., the proportion of incorrect predictions out of total number of predictions. Since we consider the $0-1$ loss function, the misclassification rate is essentially an estimator of the expected prediction error (EPE).

The experiments consist of four scenarios that correspond to four levels of queueing information. Scenario I is our baseline scenario, for which the feature vector includes only the skill group of the resource without further queueing information. Scenario II considers queue lengths of the three queues (low-priority, regular and VIP) as additional features. In Scenario III, the queue length is replaced by the waiting time of the head-of-line (HOL) for each queue. Lastly, Scenario IV includes all the above, namely, skill, queue-length and HOL waiting time.

To run the experiments, we first derived the allocation decisions and feature vectors are described in Section 4. For each experiment iteration, we randomly divided the allocation decisions into two subsets: a training set (75% of the data set) and a test set (25%), which is common practice when performing statistical model assessment [13, Ch. 8]. During each iteration the controlled variables were altered, while the misclassification rate was measured. We repeated the process of dividing the data set and running the experiment for 10 times. We used the implementations of LDA, MLR, decision trees and random forests provided by R[4]. For the decision tree algorithm we used the cross-entropy method. For random forests, in order to limit the complexity of the algorithm, we relied on 10 trees, a node size of 50, and disabled recalculations of the proximity matrix.

Results. The results of our experiments are depicted in Fig. 6, which plots the achieved misclassification rate for the six methods: four data mining methods (LDA, MLR, decision trees and random forests) and two queueing heuristics (Longest-Queue-First (LQF) and Most-Delayed-First (MDF)). For each of the data mining methods, we have four results, one for each type of queueing information (i.e., skill, queue-length, head-of-line, all).

We observe that for the baseline scenario, when the only available information is the skill group of the resource, all data mining algorithms yield the same misclassification rate (of 37%). In Scenarios II-IV, where queueing information is introduced, LDA does not improve beyond the baseline scenario. MLR improves by 8% when queue-length is considered in the prediction. However, this is generally inferior compared to decision

[4] http://www.r-project.org/

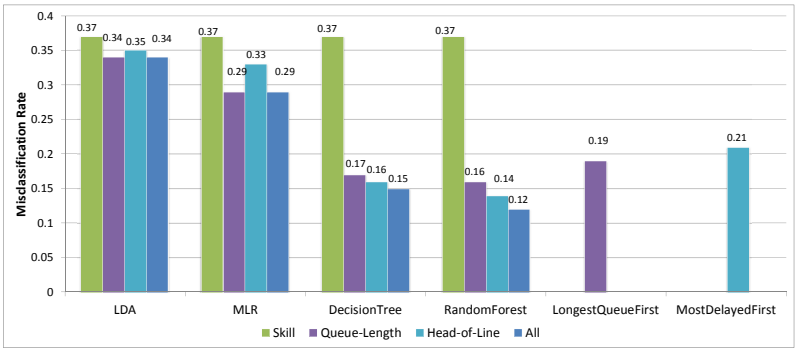

Fig. 6: Misclassification rates of the discovered protocols

trees, random forests, and the Longest-Queue-First heuristic that scored 17%, 16% and 19% misclassification rate, respectively. Considering the influence of the HOL waiting time, none of the linear classifiers (LDA and MLR) improves, whereas decision trees, random forests, and the Most-Delayed-First heuristic achieve 16%, 14% and 21% misclassification. Decision trees and random forests further improve slightly when all types of queueing information is considered.

Discussion. First and foremost, we observe that the linear data mining techniques yield comparably high misclassification rate values. This can be explained by their strong assumptions. For instance, the LDA assumes Gaussian densities of the feature vector conditioned on decisions. As a consequence, the linear methods impose a low computational effort and, when the respective assumptions hold true, provide a precise classification [13, Ch. 4]. However, due to their assumptions, the methods are not applicable to any scenario and, thus, lead to poor performance for our data set.

In contrast, decision trees and random forests do not impose assumptions on the feature vector and therefore, are more robust to various distributions of these vectors. On the downside, tree methods are based on greedy algorithms, which renders it unlikely that they converge to an optimal splitting of the feature vector space. Despite this shortcoming, the tree-based techniques yield the best prediction results in our comparative analysis. In addition, they also allow for decrypting complex scheduling protocols from event data as we exemplified it in Section 4.

Queueing heuristics perform surprisingly well, although their underlying assumption do not hold throughout large portions of the data: the call center is not constantly in heavy-traffic and the three queues do not have the same priorities. The biggest advantage of queueing heuristics is their simplicity; to apply them, one does not need to learn from data or use sophisticated black-box techniques. The information that they require is readily available in any service operation environment, so that they can be easily programmed into any recommendation system. However, our results also indicate that the misclassification rate is not on the level of tree-based methods.

Our results raise the question whether queueing heuristics can be improved, for instance, by incorporating priorities between the various queues. Instead of selecting the longest queue first, or the customer from the most delayed queue, we may consider

attaching weights to each of the queues. Suppose that a queue-length vector, $\boldsymbol{q}(t) = (q_{d_1}(t), \ldots, q_{d_n}(t))$, receives corresponding weights, $w_1, \ldots, w_n$ with $\sum_1^n w_i = 1$. Then, with F as the set of feasible allocation decisions at time t, the predicted decision could be defined as:

$$d = \arg\max_{f \in F} q_f(t) \cdot w_f. \tag{10}$$

In fact, this protocol corresponds to the heavy-traffic rule of Fixed-Queue-Ratio (FQR) [15]. The downside of such an extension is that it requires a learning phase for $w_1, \ldots, w_n$, which would inflict the aforementioned advantage of queueing heuristics.

6 Related Work

The work presented in this paper fits under the umbrella of context-aware, operational process mining. Process mining research has seen a remarkable surge lately, providing techniques for the discovery of process-related models from event data, see [11] for a broad overview. Recently, the importance of context information for process mining was highlighted [17]. Our work follows this line by arguing that the most narrow scope, i.e., the context of process instances, is not sufficient for operational analysis of a process. The behavior of resources and allocation protocols as mined in our work are part of the broader process context, beyond single instances.

Operational process mining refers to the creation of models for quantitative analysis. Here, evaluation of temporal properties has received considerable attention, for example, the prediction of processing delays or completion times for running cases by constructing simulation models from event data. In [18], time prediction is grounded on a Coloured Petri net comprising resource and timing information. Other work enriches such predictions with stochastic information [12]. Time prediction based on abstractions of states and state transitions was developed in [5]. It has been argued though, that realistic time prediction requires modeling resource utilization appropriately [19, 20]. In [19], the authors used regression analysis to show that speed of service is indeed affected by the workload. To take such context-factors into account, time prediction based on abstractions of states and state transitions was recently extended to consider context information such as system load [6]. We argue that, for service processes, consideration of the system load is not sufficient to explain all delays of processing. Our work highlights the importance of the interplay of customers and resources for performance analysis, and the relevance of queueing information and scheduling protocols in particular.

The application of decision trees to extract resource scheduling rules was also discussed by Li and Olafsson [26], yet only applied to simulated event logs. In contrast, our experiments were conducted on a real-world data set and used a variety of statistical learning and queue mining techniques that go beyond decision trees, which are considered state-of-the-art in mining of scheduling rules.

Further, the duality between a decision problems in processes and classification problems was leveraged for mining of branching conditions [22, 23] (falling into the process instance context). Our work relies on similar techniques, but exploits queueing information and works on the broader context of processes to discover protocols.

Despite its importance for performance analysis, only few works analyze behavior of resources. In [21, Ch .2], resource (aka server) networks were defined as directed graphs that depict resource-flow through service activities or customer queues, which can be used to estimate resource absenteeism rates. These rates, in turn, allow for long and short-term workforce planning in service processes. Our work supports this approach by discovering scheduling protocols from event data.

The presented queueing heuristics are grounded in queueing control theory for heavy-traffic scenarios, cf., [16, 15]. Given a certain structure of queues and service providers, control theory gives rise to protocols that are provable optimal as the demand reaches system capacity. The two presented heuristics are particularly inspired by delay predictors that do not assume steady-state, but work on wait time information of the current snapshot of a system [24, 25]. We follow the same idea when exploiting only the current state of queues instead of historical data.

7 Conclusion

In this paper, we argued that for performance analysis of service processes and estimation of processing delays, it is crucial to understand the resource scheduling protocols that match customers with service providers. Given that in many cases, service processes are supported by information systems that track the execution of activities by customers as well as the behavior of resources, we advocate the discovery of such protocols from event data. Following the hypothesis that queueing information serves as an essential element in mining scheduling protocols, we presented two specific types of mining techniques. First, we showed how classification methods from data mining can be used when including queueing information in their explanatory features. Second, we proposed heuristics that originate in queueing theory and exploit solely the current state of a system. We tested both types of techniques using a large real-world data set from the telecommunications sector. Our results indicate that data mining with decision trees and random forests is able to derive predictors for scheduling decisions with up to 88% precision. In addition, queueing heuristics also perform well reaching levels of up to 81% precision. We conclude that high prediction precision can be achieved already with online methods that do not require a preliminary learning phase on historic data.

As part of future work, we aim at developing and evaluating queueing heuristics that are enriched by a small set of features extracted from historical data and, therefore, allow for a different trade-off of prediction effort and precision. Also, such heuristics should be extended symmetrically to the customer's perspective: upon the arrival of a customer service request, the presence of a set of free resources of various types implies the need to take a routing decision.

References

1. Fitzsimmons, J.A., Fitzsimmons, M.J.: Service Management: Operations, Strategy, Information technology. McGraw-Hill/Irwin Boston (2004)
2. Gans, N., Koole, G., Mandelbaum, A.: Telephone call centers: Tutorial, review, and research prospects. Manufacturing & Service Operations Management **5**(2) (2003) 79–141

3. Buzacott, J.A., Shanthikumar, J.G.: Stochastic Models of Manufacturing Systems. Prentice Hall, Englewood Cliffs, NJ (1993)
4. Hall, R.W.: Queueing Methods: For Services and Manufacturing. Prentice Hall, Englewood Cliffs NJ (1991)
5. van der Aalst, W.M., Schonenberg, M., Song, M.: Time prediction based on process mining. Information Systems **36**(2) (2011) 450–475
6. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: OTM Conferences (1). Volume 7565 of Lecture Notes in Computer Science., Springer (2012) 287–304
7. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining - predicting delays in service processes. In: Proceedings of the 26th International Conference on Advanced Information Systems Engineering (CAiSE'14), Thessaloniki, Greece. (2014) To appear.
8. Decker, G., Weske, M.: Interaction-centric modeling of process choreographies. Inf. Syst. **36**(2) (2011) 292–312
9. Object Management Group: Business Process Model and Notation (BPMN) 2.0. (2011)
10. Garnett, O., Mandelbaum, A.: An introduction to skills-based routing and its operational complexities. Teaching notes (2000)
11. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
12. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In Basu, S., Pautasso, C., Zhang, L., Fu, X., eds.: ICSOC. Volume 8274 of Lecture Notes in Computer Science., Springer (2013) 389–403
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA (2001)
14. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
15. Gurvich, I., Whitt, W.: Service-level differentiation in many-server service systems via queue-ratio routing. Operations Research **58**(2) (2010) 316–328
16. Gurvich, I., Whitt, W.: Scheduling flexible servers with convex delay costs in many-server service systems. Manufacturing & Service Operations Management **11**(2) (2009) 237–253
17. van der Aalst, W., Dustdar, S.: Process mining put into context. IEEE Internet Computing **16**(1) (2012) 82–86
18. van der Aalst, W., Nakatumba, J., Rozinat, A., Russell, N.: Business process simulation: How to get it right. BPM Center Report BPM-08-07, BPMcenter. org (2008)
19. Nakatumba, J., van der Aalst, W.M.P.: Analyzing resource behavior using process mining. In Rinderle-Ma, S., Sadiq, S.W., Leymann, F., eds.: Business Process Management Workshops. Volume 43 of Lecture Notes in Business Information Processing., Springer (2009) 69–80
20. Nakatumba, J.: Resource-Aware Business Process Management: Analysis and Support. PhD thesis, Technische Universiteit Eindhoven, Eindhoven (12 2013)
21. Senderovich, A.: Multi-Level Workforce Planning in Call Centers. Master's thesis, Technion (2012)
22. Rozinat, A., van der Aalst, W.M.P.: Decision mining in prom. In Dustdar, S., Fiadeiro, J.L., Sheth, A.P., eds.: Business Process Management. Volume 4102 of Lecture Notes in Computer Science., Springer (2006) 420–425
23. de Leoni, M., Dumas, M., García-Bañuelos, L.: Discovering branching conditions from business process execution logs. In Cortellessa, V., Varró, D., eds.: FASE. Volume 7793 of Lecture Notes in Computer Science., Springer (2013) 114–129
24. Whitt, W.: Predicting queueing delays. Management Science **45**(6) (1999) 870–888
25. Ibrahim, R., Whitt, W.: Real-time delay estimation based on delay history. Manufacturing and Service Operations Management **11**(3) (2009) 397–415
26. Li, Xiaonan, and Sigurdur Olafsson: Discovering dispatching rules using data mining. Journal of Scheduling **8**(6) (2005) 515–527