Learning to Predict User Replies in Interactive Job Scheduling *

Johannes Varga[1][0000−0003−1413−7115], Günther Raidl[1][0000−0002−3293−177X], and Tobias Rodeman[2][0000−0001−6256−0060]

[1] Algorithms and Complexity Group, TU Wien, Austria
{jvarga,raidl}@ac.tuwien.ac.at
[2] Honda Research Institute Europe GmbH, Germany
tobias.rodemann@honda-ri.de

Abstract. We consider a learning task that arises within an interactive job scheduling setting, in which a scheduler plans the execution of jobs that require the presence of human users. Availabilities of these users shall be considered but are only known partially, and thus the scheduler presents queries to the users to receive more information about the users' availabilities. The replies of the users help creating a better schedule. Having a precise understanding of typical user behavior is crucial for the efficacy of the scheduler. As the scheduling problem must be solved repeatedly over time, e.g., weekly, the knowledge gained from previous instances can be used to learn a user model. In this work we employ Bayesian Learning and investigate three different models to predict user replies to queries and train them by means of the framework of Probabilistic Programming. Two models learn time-independent, respectively time-dependent, probabilities for a user to either become available, stay available, become unavailable, or stay unavailable from one timestep to the next. The third model learns time intervals in which the user is available with normally distributed endpoints. These models are experimentally evaluated and compared on two datasets, one based on artificially generated user availabilities, the other on real-world data. Results show that especially the time-dependent model performs well and near-optimal for the artificial dataset while the time interval model works best on the other dataset.

Keywords: Job Scheduling · Bayesian Learning · Interactive Optimization

1 Introduction

Coordinating access to central resources shared between human users is a challenging task. Besides operator interests such as minimizing costs for making the resources available, restrictions of the users such as availability times have to be taken into account when creating a schedule for the usage of the resources. While

* J. Varga acknowledges the financial support from Honda Research Institute Europe.

constraints on the side of the resources and operator as well as the cost structure are usually known, detailed availability times of the users are frequently unknown and too burdensome and impractical for every user to completely specify upfront. We consider the case where it is important to have a good understanding of the users' availability times in order to find an effective and cost-efficient schedule. For this purpose, we allow the scheduling framework to interact with the users by making interactive queries to them to obtain more information about their availabilities. The amount of this interaction, however, should be kept to a minimum to avoid annoying the users, and queries should be chosen wisely to obtain most relevant information towards realizing a cost-efficient feasible schedule.

More specifically, the scheduler uses a model of the users' general patterns of availability and interacts with the users to also account for deviations from the general patterns. Aim of this work is to learn different behavioral models for the users, based on the information on users collected over time through interaction.

We assume a discrete time horizon of multiple separate days, and we will refer to these days altogether as week. There are multiple users and their access to shared resources has to be coordinated. In general, users are not available throughout the whole time horizon and their availabilities have to be respected. To make the process of coordinating access to the resources as smooth as possible from the user perspective, users initially just make single suggestions for time intervals in which they want to use the resources. The scheduler tries to find a first feasible schedule and then works towards further improving it by querying the users for more information about their availabilities. The replies of the users are thus used to update the scheduler's knowledge about user availabilities. To conform with the literature on scheduling problems, we will refer to the resources as machines and to the usage requirements of the users as jobs.

Having a general understanding of the users' availabilities is essential to compile meaningful queries and avoid stumbling about in the dark. The common patterns can be different depending on the application-context and using a predetermined and fixed user model to predict user availabilities is therefore not reasonable. We therefore propose to learn an availability model based on the information about the users obtained from their replies to queries in previous weeks. More specifically, we will investigate different models to learn and predict user availabilities. As a baseline we use a simple Markov model with time-independent transition probabilities. Additionally, we propose a Markov model with time-dependent transition probabilities and a model that considers availability intervals with normally distributed endpoints. To address the peculiarities of the learning task, we use Bayesian Learning and the flexible framework of Probabilistic Programming and develop a quickly converging algorithm within this framework to infer parameters for each model.

Our main contributions are (a) to formulate the new learning and prediction task from the setting described above and to publish two associated datasets, (b) to propose two new probabilistic models for representing user availabilities and (c) to develop a quickly converging algorithm that trains the two new models and the former simpler model from the literature. We compare the models on two

datasets that are derived from prior work, one is based on randomly generated user availabilities, the other on user availabilities derived from the Dutch time-use-survey [19]. Results show that the time-dependent Markov model performs best and near-optimal for the first dataset with generated user availabilities, while the time interval model clearly outperforms the other models for the time-use-survey based dataset.

The remainder of this paper is structured as follows. In the next section, we summarize related work. Section 3 describes the scheduling problem and interaction procedure, and Section 4 formalizes and discusses the prediction task. Afterwards, the models are presented in Section 5 and the inference of the model parameters is discussed in Section 6. Finally, Section 7 compares the models with each other and Section 8 concludes the work.

2 Related Work

Varga et al. [20,21] already considered the described scheduling scenario and proposed an approach to identify the most meaningful queries, which builds upon a user model that predicts acceptance probabilities of potential queries in a simple fashion without learning over time. Queries with an acceptance probability below some threshold are filtered out, and the scheduler selects those from the remaining queries that improve the objective the most when accepted. They compare a rather simple user model, a two-state Markov chain with time-independent transition probabilities, with the true user model that was used to generate the (artificial) benchmark instances and with using no user model at all. Conclusions are that the schedule can be significantly improved within five rounds of user interaction, and that the true user model leads to significantly faster convergence. Therefore, the authors conjectured that a *learned* model will likely lead to substantial improvements if the true model is not known.

Learning of user behavior in the temporal domain has been done in the context of calendar scheduling. There, a program called the calendar scheduling agent supports a user in arranging meetings with other humans and to better do so, it learns the user's preferences and habits. Machine learning models such as decision trees [16], the weighted-majority algorithm or the Winnow algorithm [2] have been applied to the learning task. These approaches make predictions based on information that is not available in our setting, such as the attendees of the meeting. Therefore they cannot be applied for our learning task.

Interruption management is another line of research where user presence is predicted in the temporal domain. It is concerned with determining times to interact with users that suits them well, considering their current tasks and availability. Most relevant for our work, Horvitz [10] presents a system that utilizes a Bayesian network to predict and forecast a users attendance and interruptibility given their calendar and other observations, such as desktop activity. Similarly, Horvitz and Apacible [11] use a Bayesian network to predict the current attention and cost of interruption based on the users calendar, prior interaction with different units and information gathered from the computers microphone and

camera. Horvitz et al. [12] learn personalized models for the cost of interruption based on similar information, and Kapoor and Horvitz [14] extend this work to online learning, carefully selecting requests for user feedback based on the predicted value and costs of interaction.

Ignoring the user interaction aspects for now, we consider a scheduling problem with related machines and time- and machine-dependent costs for using a machine, see also the formal definition in the next section. Wang et al. [22] solve a similar problem that further takes into account the makespan, but does not consider users with availabilities, with a problem specific heuristic, a genetic algorithm and a Mixed Integer Linear Programming (MILP) formulation. The same problem has been approached by Anghinolfi et al. [1] with a greedy heuristic, local search and a MILP model. For a generalization of our problem, when neglecting the user aspect, Ding et al. [6] proposed a MILP formulation, which has been improved by Cheng et al. [4] and Saberi et al. [17].

Bayesian Learning [18] has received a lot of attention in the last years. It is based on Bayes rule to update the prior beliefs, given as probability distribution over a parameter space, to obtain the posterior beliefs on the parameters. Applications appear in many disciplines, such as social behavioral sciences [3] and ecology [13]. Prominent subfields include Bayesian networks, Bayesian deep learning, and Bayesian optimization. Strengths of Bayesian Learning are its flexibility, data-efficiency and inherent immunity to overfitting. In this work, we use probabilistic programming [15] to describe the underlying model and a Markov Chain Monte Carlo algorithm [5] to infer parameters. Markov Chain Monte Carlo methods sample new parameter values based on previous parameter values and generate a number of parameter sets that represent the posterior distribution. Metropolis Hastings [9] is an algorithm that is often used in conjunction with Markov Chain Monte Carlo methods, and it ensures that a proposal distribution for new parameter values fulfills the necessary conditions by computing an acceptance probability and only accepting new values with this probability.

3 Scheduling Setting

We consider a basic scheduling problem with a discrete time horizon $T = \{(t^{\text{day}}, t) \mid t^{\text{day}} = 1, \dots, t^{\text{max-day}}, t = 1, \dots, t^{\text{max}}\}$ of $t^{\text{max-day}}$ days, each with t^{max} discrete timesteps. Denote the set of users with U, the set of jobs of user $u \in U$ with J_u and the set of all jobs with $J := \bigcup_{u \in U} J_u$. Each job $j \in J$ has a duration $d_j \in \{1, \dots, t^{\text{max}}\}$ and is not allowed to span multiple days. The set of machines is given by M. Each job $j \in J_u$ requires exclusive access to a machine $i \in M$ and the availability of its user u while it is performed, and it must not be interrupted once started. Performing a job on machine $i \in M$ at timestep $t \in T$ induces costs $c_{it} \geq 0$. It is not required to schedule all jobs and not scheduling job $j \in J$ comes with a penalty $q_j \geq 0$. The objective is to minimize the total costs including the penalties for unscheduled jobs.

In general, users are not available throughout the whole time horizon and jobs need to be scheduled in a way that respects the users' availabilities. To

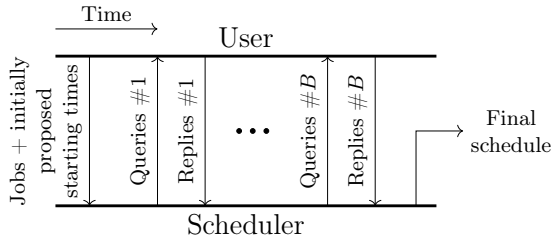

Fig. 1. Interaction protocol between the scheduler and a single user, which is performed in parallel for each user.

make the process of scheduling the jobs as smooth as possible from the user perspective, users are not required to fully specify their availabilities. Instead, the following interaction protocol between the scheduler and each users is used, see also Figure 1. First, each user submits the jobs they want to be scheduled along with a proposed starting time for each of their jobs, to the scheduler. When considering the job durations, these starting times result in time intervals for which the user confirms to be available; we refer to these time intervals as the *proposed intervals*. Next, the scheduler interacts with the users to obtain more information about their availabilities that is most relevant to possibly improve an initially determined schedule. This is done for B rounds, and in each round the scheduler first computes a set of most meaningful queries, which are then relayed to the users. On average, each user gets one query in each round, although one user might get multiple queries in a round if other users do not get any queries in the same round. A query is a time interval and the user either accepts it, if they are available in the whole interval, or rejects it, if they are (partly) unavailable. These *accepted* and *rejected intervals* are used to update the schedulers knowledge about the users' availabilities and allows it to determine a possibly improved schedule and to prepare the queries for the next round. Alongside the set of proposed intervals, these accepted and rejected intervals give insight into the users behavior, and we will use them to train our models.

4 Prediction Task

Proposed, accepted and rejected intervals of each user are collected over the course of multiple weeks and used to train our models. The aim is to predict user replies for the current week after the scheduler already interacted with the users. Thus, a training sample $s^{\text{train}} = (I^{\text{prop}}, I^{\text{acc}}, I^{\text{rej}})$ consists of the sets of initially proposed, accepted and rejected time intervals of a previous week. Each training sample gives clues on the distribution of the user's availabilities and they are used to train a user model. Test samples $s^{\text{test}} = (I^{\text{prop}}, I^{\text{acc}}, I^{\text{rej}}, I^{\text{pred}}, \hat{I}^{\text{pred}})$ carry another set I^{pred} of intervals that are potential queries and associated labels $\hat{I}^{\text{pred}} : I^{\text{pred}} \to \{\textbf{false}, \textbf{true}\}$ representing the replies the users would give to these queries. The task is to predict for each potential query, whether the

user will accept it. To evaluate the model, its prediction is compared to the label of the query. Note that also test samples include information from the user interaction of previous rounds (I^{prop}, I^{acc} and I^{rej}), and this information should also be exploited to make a prediction. Further note that we determine the labels in our simulations based on true user availabilities solely to evaluate the models; they are never available to the scheduler.

Our prediction task is different from typical machine learning tasks in multiple aspects. For once, training samples carry uncertainty, since the true user availabilities are not given. This also means that there are no dedicated labels that could be used for training by the scheduler since the users never specify their full availabilities to the scheduler. Furthermore, the information contained in the samples has some structure. Accepted intervals imply that the user is available in these whole intervals, while rejected intervals imply that the user is not available in at least one timestep of each such interval. Initially provided intervals resemble accepted intervals, and thus there is a bias towards accepted intervals. In general, we assume that the duration of timesteps is chosen short enough to avoid repeated changes of a user's availability within a few timesteps, otherwise the time horizon is chosen too coarse to sufficiently capture the details of the user's availabilities. Furthermore, the scheduler should be able to make good predictions already after running for a few weeks when only few training samples, in the range from tens to few hundreds, are available. This means that as much information as possible has to be exploited from the samples and the scenario to obtain reasonable results. For these reasons, it is not straightforward to apply classical machine learning models, such as Support Vector Machines. Instead, we rely on custom models and infer their parameters with Probabilistic Programming to optimally exploit all available information contained in the few training samples. These models are described in the subsequent section.

5 Models

In the following, we present three models for the user availability prediction task. Since there are only a few training samples for each user, some correlation between the different users' availabilities has to be assumed to learn a reasonable model. We assume the availabilities of different users on different days to be all identically distributed for the simplicity of the presentation and since we do not have data available to represent and test user- and day-specific behavior. Thus we only learn a single model for all users and all days.

5.1 Time-(in)dependent Markov Model

The first – very simple – model only features two parameters and we will refer to it as time-independent Markov model and use it as a baseline model to compare our other models to. It has been used by Varga et al. [20,21] to predict user replies, though with fixed, non-learned parameters. The model is based on a Markov chain with the two states 0 and 1, representing the user being unavailable

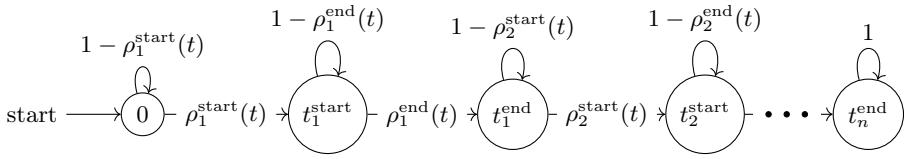

Fig. 2. Markov chain for the interval model.

in the current timestep or available, respectively. Before the first timestep, the user is assumed to be unavailable and, only depending on the state of the current timestep, they become available with a probability ρ_{01}, become unavailable with a probability ρ_{10}, and stay unavailable or available with the complementary probabilities $1-\rho_{01}$ and $1-\rho_{10}$. Similar to assuming the user to be unavailable before the start of the day, we also assume that the user is unavailable after the end of the day by extending a day by one timestep and imposing the condition that the user is unavailable in this timestep. The transition probabilities ρ_{01} and ρ_{10} are the learnable parameters of this model. Using beta distributions as prior distributions for these parameters seems natural, since they represent probabilities with values in $[0,1]$. As we expect that there are no repeated changes in the availability within a few timesteps, parameters ρ_{01} and ρ_{10} should be less than 0.5, and therefore we choose the parameters $\alpha = 1$ and $\beta = 3$ for the beta distribution, leading to an the expected value 0.25, which is centered between 0 and 0.5.

The time-dependent Markov model is identical to the time-independent Markov model, except that the transition probabilities $\rho_{01}(t)$ and $\rho_{10}(t)$ depend on the timestep t. This increases the number of parameters to two per timestep and allows the model to learn at which timesteps the users are available more likely. Different to before, the assumptions that the user is unavailable before the first and after the last timestep do not make a difference and are not needed as the corresponding probabilities can be learned individually.

5.2 Time Interval Model

In our third model, we again assume that the user's availability does not change frequently over few successive timesteps and therefore the user will only be available in few intervals throughout the day. We model this for up to n intervals within the day, whose endpoints follow rounded normal distributions, where the mean values μ_i^{start}, μ_i^{end} and standard deviations σ_i^{start}, σ_i^{end}, $i = 1, \ldots, n$ are learnable parameters. Note that the number of intervals n is here a fixed strategy parameter and is not learned. To break symmetries, we impose the condition that these intervals have to be ordered, that each interval has a duration of at least one, and that there is at least one timestep between two intervals. I.e., we consider intervals $[t_i^{\text{start}}, t_i^{\text{end}}]$ with $t_i^{\text{start}} \sim \texttt{Round}(\mathcal{N}(\mu_i^{\text{start}}, \sigma_i^{\text{start}}))$, $t_i^{\text{end}} \sim \texttt{Round}(\mathcal{N}(\mu_i^{\text{end}}, \sigma_i^{\text{end}}))$, $i = 1, \ldots, n$, and where $t_1^{\text{start}} < t_1^{\text{end}} + 1 < t_2^{\text{start}} < \ldots < t_n^{\text{end}} + 1$; $\mathcal{N}(\cdot,\cdot)$ is the normal distribution, $\texttt{Round}(\cdot)$ rounds to the

nearest integer in $[1, t^{\max}]$. The expected values μ_i^{start} and μ_i^{end} and standard deviations σ_i^{start} and σ_i^{end} are parameters to the model that are learned.

This model can be formulated with the Markov chain visualized in Figure 2 and this formulation will be important for learning the model parameters. There is an initial state 0 in which the user is not available and for each interval, there is a state t_i^{start} in which the user is available, and a state t_i^{end} in which the user is not available and all states have a predefined order in which they have to be visited. We use the junction tree algorithm to compute the conditional probabilities $p_i^{\text{start}}(t) = \mathbb{P}(t_i^{\text{start}} = t+1 \mid t_1^{\text{start}} < t_1^{\text{end}} + 1 < \ldots)$ and $p_i^{\text{end}}(t) = \mathbb{P}(t_i^{\text{end}} = t \mid t_1^{\text{start}} < t_1^{\text{end}} + 1 < \ldots)$ since the junction tree algorithm delivers exact probabilities and is efficient in this case. The transition probabilities are then further derived as

$$\rho_i^{\text{start}}(t) = \mathbb{P}(t_i^{\text{start}} = t+1 \mid t_i^{\text{start}} \geq t+1, t_1^{\text{start}} < t_1^{\text{end}} + 1 < ...) = \frac{p_i^{\text{start}}(t)}{\sum_{t' \geq t} p_i^{\text{start}}(t')} \tag{1}$$

$$\rho_i^{\text{end}}(t) = \frac{p_i^{\text{end}}(t)}{\sum_{t' \geq t} p_i^{\text{end}}(t')}. \tag{2}$$

6 Bayesian Inference

To infer parameters for our models, given a set S^{train} of training samples, we use Bayesian Learning [18]. Advantages are its broad applicability with the capability to model and take into account all the specific aspects of our setting and that it also provides a measure for the uncertainty of the result by not just giving a single value for each parameter, but multiple sets of model parameters. When using one of the models as part of the scheduler to actively learn user behavior over the course of multiple weeks, the uncertainty measure will help to select queries that improve the knowledge about the users. Foundation of Bayesian Learning is Bayes rule, which states that the posterior distribution $\mathbb{P}(\theta \mid D)$ of the parameters θ is proportional to their prior distribution $\mathbb{P}(\theta)$ and the likelihood, which is the probability $\mathbb{P}(D \mid \theta)$ of seeing the observed data D:

$$\mathbb{P}(\theta \mid D) \propto \mathbb{P}(D \mid \theta)\mathbb{P}(\theta) \tag{3}$$

Probabilistic Programs can be used to specify a model and are ordinary programs, extended by the statements **smpl** and **obs**, which sample parameters from a specified prior distribution and observe that certain conditions are fulfilled, e.g. that the program generated the seen data. Prior probability and likelihood can be computed from a probabilistic program and parameter values θ.

Algorithm 1 shows the skeleton of the probabilistic program that we use in conjunction with two model-specific functions `Prior(Model)` and `Model(`θ`)`, which sample the model parameters from their prior distribution and sample a set of availabilities based on these model parameters, respectively, and which are discussed below for each of our models. The probabilistic program first samples the model parameters and then, for each training sample s^{train}, samples a set of availabilities and observes that it is compatible with s^{train}. To account for the

Algorithm 1: Model-independent skeleton of the probabilistic program to condition on training samples S^{train}.

Input: Training samples S^{train}
1 **smpl** $\theta \sim$ `Prior(Model)`
2 **for** $(I^{\text{prop}}, I^{\text{acc}}, I^{\text{rej}})$ **in** S^{train} **do**
3 **smpl** $T^{\text{avail}*} \sim$ `Model(`θ`)`
4 **for** $[t_1, t_2]$ **in** I^{prop} **do**
5 $I \leftarrow$ `Ints(`$T^{\text{avail}*}$`,` $t_2 - t_1 + 1$`)`
6 **smpl** $[t_1', t_2'] \sim$ `Uniform(`I`)`
7 **obs** $t_1' = t_1$ *and* $t_2' = t_2$
8 **obs** $[t_1, t_2] \subseteq T^{\text{avail}*} \quad \forall [t_1, t_2] \in I^{\text{acc}}$
9 **obs** $[t_1, t_2] \nsubseteq T^{\text{avail}*} \quad \forall [t_1, t_2] \in I^{\text{rej}}$

Algorithm 2: Model-independent part of the sampling procedure.

Input: Training samples $\{s_1^{\text{train}}, \ldots, s_{|S^{\text{train}}|}^{\text{train}}\}$, $n \in \mathbb{N}$
Output: Parameter sets $\{\theta_1, \ldots, \theta_n\}$
1 $\theta \leftarrow$ `InitParameters(Model)`
2 $T_k^{\text{avail}*} \leftarrow T \qquad \forall k \in \{1, \ldots, |S^{\text{train}}|\}$
3 **for** j **in** $\{1, \ldots, n\}$ **do**
4 **for** k **in** $\{1, \ldots, |S^{\text{train}}|\}$ **do**
5 $G \leftarrow$ `Prob-graph(`θ`,` s_k^{train}`)`
6 $\tilde{T}^{\text{avail}} \leftarrow$ `RandomAvail(`G`)`
7 **if** `Rand(`$[0, 1)$`)` $<$ `Acc-Prob` **then**
8 $T_k^{\text{avail}*} \leftarrow \tilde{T}^{\text{avail}}$
9 $\theta \leftarrow$ `SampleParameters(`θ`,` $T^{\text{avail}*}$`)`
10 $\theta_k \leftarrow \theta$
11 **return** $\{\theta_1, \ldots, \theta_n\}$

bias towards the user being available caused by proposed intervals, the process of proposing an interval is modelled in lines 5 to 7 by assuming that the user selects one of the possible intervals of the given length uniformly at random. The function `Ints(`T`,` d`)` returns all intervals of duration d that are contained in the discrete set T of timesteps. This program assigns each tuple of values of sampled variables a probability by multiplying the probabilities or probability densities of sampling exactly those values and multiplying with 0 if an observe statement is not fulfilled. These products are then proportional to the posterior distribution of the sampled variables and, when marginalized, to the posterior distribution of the parameters θ, which we are interested in. Note that we perform these computations with probabilities, as is usually done in probabilistic programming, in the space of logarithmic values, where these products turn into sums and there is no risk of an arithmetic underflow.

To sample from the posterior distribution, we alternate between sampling new sets of availabilities (lines 4-8) and new parameters while fixing the other, also referred to as Gibbs sampling [8], see Algorithm 2. Availabilities and parameters can be sampled individually with distributions that approximate their posterior distribution well and therefore doing Gibbs sampling in this way in conjunction with Metropolis Hastings to compensate the approximation error leads to fast convergence to the overall posterior distribution. Sampling the parameters is model-specific and discussed below for each of the models. Sets of availabilities are generated individually for each training sample using a datastructure described by Varga et al. [21] and we will refer to this datastructure as probability graph. A probability graph can be created based on the Markov chain of the model; it represents the probability distribution of availabilities according to the model after observations, such as accepted or rejected queries, have been made. Note that in line 5, G represents the probability graphs of all days within the considered week. Since the probability graph treats proposed exactly like

Algorithm 3: Probabilistic program for the Markov Models.

Input: p_{01}, p_{10}
Output: $T^{\text{avail}*}$

1 $T^{\text{avail}*} \leftarrow \{\}$
2 **for** t^{day} **in** $\{1, \ldots, t^{\text{max-day}}\}$ **do**
3 avail $\leftarrow$ **false**
4 **for** t **in** $\{1, \ldots, t^{\max} + 1\}$ **do**
5 $p \leftarrow$ $1 - p_{10}(t)$ **if** avail **else** $p_{01}(t)$
6 avail $\leftarrow$ **smpl** `Bernoulli(`p`)`
7 **if** *avail* **then**
8 $T^{\text{avail}*} \leftarrow T^{\text{avail}*} \cup \{(t^{\text{day}}, t)\}$
9 **obs** $\neg$ avail
10 **return** $T^{\text{avail}*}$

Algorithm 4: Probabilistic program for the interval model.

Input: n, μ_i^{start}, σ_i^{start}, μ_i^{end}, σ_i^{end}, $i = 1, \ldots, n$
Output: $T^{\text{avail}*}$

1 $T^{\text{avail}*} \leftarrow \{\}$
2 **for** t^{day} **in** $\{1, \ldots, t^{\text{max-day}}\}$ **do**
3 **for** i **in** $\{1, \ldots, n\}$ **do**
4 $t_i^{\text{start}} \leftarrow$ `Round(`**smpl** $\mathcal{N}(\mu_i^{\text{start}}, \sigma_i^{\text{start}})$`)`
5 $t_i^{\text{end}} \leftarrow$ `Round(`**smpl** $\mathcal{N}(\mu_i^{\text{end}}, \sigma_i^{\text{end}})$`)`
6 **obs** $t_i^{\text{start}} \leq t_i^{\text{end}} \quad \forall i \in \{1, \ldots, n\}$
7 **obs** $t_i^{\text{end}} + 1 < t_{i+1}^{\text{start}} \quad \forall i \in \{1, \ldots, n-1\}$
8 $T^{\text{avail}*} \leftarrow$ $T^{\text{avail}*} \cup \bigcup_{i=1}^{n} [(t^{\text{day}}, t_i^{\text{start}}), (t^{\text{day}}, t_i^{\text{end}})]$
9 **return** $T^{\text{avail}*}$

accepted intervals, we have to take into account Line 6 of Algorithm 1 to get the correct posterior distribution. We use Metropolis Hastings [9] for this correction, i.e., the newly drawn sample is kept with probability

$$\texttt{Acc-Prob} = \min\left(1, \frac{\mathbb{P}(\tilde{T}^{\text{avail}})}{\mathbb{P}(T_k^{\text{avail}*})}\right) = \min\left(1, \prod_{[t_1,t_2]\in I^{\text{prop}}} \frac{|\texttt{Ints}(T_k^{\text{avail}*},\, t_2 - t_1 + 1)|}{|\texttt{Ints}(\tilde{T}^{\text{avail}},\, t_2 - t_1 + 1)|}\right), \quad (4)$$

since the probability to draw any element from the uniform distribution on I is $1/|I|$ and in fact this favors shorter availability intervals and therefore compensates the bias introduced by systematically having more intervals where the user is available compared to rejected intervals.

6.1 Markov Models

Algorithm 3 shows the probabilistic program to sample from the Markov models. For the time-independent version, $p_{01}(t) = p_{01}$ and $p_{10}(t) = p_{10}$ are constant functions. To sample new parameters p_{01} and p_{10} for the time-independent Markov model, observe that, since $T^{\text{avail}*}$ and thus the number of transitions is fixed, their likelihood according to the probabilistic program are proportional to

$$f_{01}(p_{01}) \propto p_{01}^{n_{01}}(1 - p_{01})^{n_{00}} \text{ and} \quad (5)$$

$$f_{10}(p_{10}) \propto p_{10}^{n_{10}}(1 - p_{10})^{n_{11}}, \quad (6)$$

where $n_{ij},\ i, j \in \{0, 1\}$ are the number of transitions from state i to state j in $T^{\text{avail}*}$. Considering the prior distribution `Beta(`*1,3*`)`, which adds a factor of $(1 - p_{01})^2$ and $(1 - p_{10})^2$, respectively, we draw new parameter values for p_{01} from `Beta(`$n_{01} + 1,\ n_{00} + 3$`)` and for p_{10} from `Beta(`$n_{10} + 1,\ n_{11} + 3$`)`. New parameter values for the time-dependent Markov model are drawn from similar beta distributions, but only counting the transitions between the two specific

timesteps of the day that correspond to that parameter. As initial values for the parameters we use 0.25 as this is the expected value of the prior distributions as established in the previous section.

6.2 Interval Model

Algorithm 4 shows the probabilistic program to generate $T^{\text{avail}*}$ for the interval model. Again, `Round()` rounds to the nearest integer in $[1, t^{\max}]$. To sample new parameter values, we use Metropolis-Hastings and derive the proposal distribution by approximating the rounded normal distribution with a continuous normal distribution. For each interval $i \in \{1, \dots, n\}$, normally distributed samples $t^{\text{start}}_{i,t^{\text{day}},k}$ and $t^{\text{end}}_{i,t^{\text{day}},k}$, $t^{\text{day}} \in \{1, \dots, t^{\text{max-day}}\}$, $k \in \{1, \dots, |S^{\text{train}}|\}$ are given and we are interested in the unknown mean values and standard deviations μ_i^{start}, σ_i^{start} and μ_i^{end}, σ_i^{end}, respectively. The variance $(\sigma_i^{\text{start}})^2$ is distributed according to a scaled inverse-χ^2 distribution with $t^{\text{max-day}}|S^{\text{train}}| - 1$ degrees of freedom and sample variance

$$\frac{\sum_{t^{\text{day}},k} \left(t^{\text{start}}_{i,t^{\text{day}},k} - \bar{t}_i^{\text{start}}\right)^2}{t^{\text{max-day}}|S^{\text{train}}| - 1} \tag{7}$$

as scaling parameter where $\bar{t}_i^{\text{start}}$ is the sample mean [7, Chapter 3]. We first draw the variance $(\sigma_i^{\text{start}})^2$ from this distribution and then draw the mean μ_i^{start} from the standard distribution $\mathcal{N}(\bar{t}_i^{\text{start}}, \sigma_i^{\text{start}}/\sqrt{t^{\text{max-day}}|S^{\text{train}}|})$. The same procedure is applied to obtain σ_i^{end} and μ_i^{end}. As prior distribution we use the uniform distribution for μ_i^{start} and μ_i^{end} and the inverse-χ^2 distribution with 0.1 degrees of freedom and scaling parameter 1 for $(\sigma_i^{\text{start}})^2$ and $(\sigma_i^{\text{end}})^2$. Note that 0 degrees of freedom would be a more uninformed choice, but Scale-inv-$\chi^2(0, 1)$ is not a proper distribution. Initial values are set to $\mu_i^{\text{start}} = \mu_i^{\text{end}} = t^{\max}/2$ and $\sigma_i^{\text{start}} = \sigma_i^{\text{end}} = t^{\max}/8$ since these values gave good results in preliminary tests.

7 Experimental Comparison

To evaluate our models, we create training and test datasets based on the instances from Varga et al. [21] with five machines, 30 users and four jobs per user. The time horizon consists of five days, each day starting at 6am and ending at 10pm, discretized into timesteps of 15min yielding a total of 64 timesteps per day. Some of these instances are based on purely synthetical user availabilities, where users are available in up to two intervals a day with rounded and capped normal distributions for the start and duration of those intervals. The other instances are based on user availabilities derived from the Dutch time-use-survey [19], considering a user to be available if they are working on-site. We assume 4 weeks of training data to be available since the scheduler is supposed be able to make good predictions already after a few weeks. To also evaluate the uncertainty of our results, we create five training sets, each based on 4 randomly selected instances of instances 1 to 20. Similarly, we generate five test datasets,

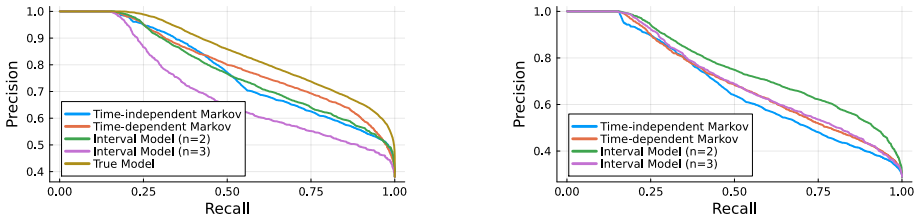

Fig. 3. Precision-recall curve of the different models, for the dataset based on generated availabilities on the left and for the dataset based on the time-use-survey on the right. Higher values for precision and recall are better.

each based on 4 randomly selected instances of instances 21 to 30. Whenever we do not report uncertainties, the results refer to a single training and a single test set. These datasets of training and test samples are generated by running the algorithm of Varga et al. [21] with the Markov approach and a probability threshold of 0.5 and collecting proposed, accepted, and rejected time intervals for each user after five rounds of interaction for the training set and after zero to five rounds of interaction for the test set. Potential queries for test samples are all possible intervals within the time horizon of job length. Training and test set are available online[3]. We compare the time-dependent Markov model and the interval model to the time-independent Markov model, which we use as baseline. Note that we are the first to consider this learning task and thus we cannot compare to approaches from the literature.

Our algorithms are implemented in Julia 1.10.5 using Gen.jl 0.4.6 as framework for probabilistic programming. All experiments were executed on a single core of an AMD Ryzen 9 5900X. For the training, we performed 1000 iterations of the Markov models, skipping the first 500 to account for convergence to the equilibrium state. All training times were below 70 seconds, prediction times for a single model and all test samples altogether below 40 seconds.

7.1 Results

We trained a model for each dataset of training samples. Each model trained on a time-use-survey based dataset is evaluated on all test datasets that are based on the time-use-survey and similarly for the datasets based on synthetical availabilities; in particular we did not investigate transfer learning capabilities here. The inference method delivers multiple parameter sets for each model. To make predictions, we use each parameter set to make a prediction and calculate the mean. Figure 3 shows the precision-recall curves of the different models for both datasets where the test samples are collected after five rounds of user interaction. Note that, for the dataset with the artificial availabilities, we do know the true model, which the user availabilities follow and the true model

[3] `https://www.ac.tuwien.ac.at/research/problem-instances/#ijsp`

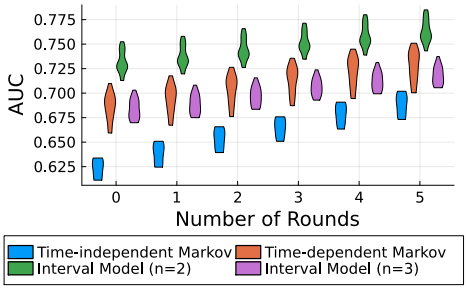

Fig. 4. Area-under-curve comparison of the precision recall plots for time-use-survey based datasets. Higher area-under-curve values are better.

represents an upper bound. As can be seen, the time-dependent Markov model is very close to the true model and was able to learn the typical user behavior very well. While the interval model with $n = 2$ intervals gives a similar performance as the time-independent Markov model, which is the baseline model, the interval model with $n = 3$ intervals performs significantly worse. We explain this by the fact that users are assumed to be available in up to two intervals and the third interval that the model has to use disturbs the position of the other two intervals.

For the dataset that is based on the time-use-survey, the interval model with $n = 2$ intervals clearly performs best and the interval model with $n = 3$ intervals performs similar to the time-dependent Markov model, both performing better than the time-independent Markov model. We explain the difference to the artificial dataset with the fact that in the artificial dataset each user has at most two intervals each day and often only one and their starting and ending times have less variance, which favors the time-dependent Markov model.

Figure 4 compares, for the time-use-survey based dataset, the area-under-curve values of the precision-recall plots for the different models, when varying the number of rounds after which test samples are collected and therefore the amount of information that is given about the user's availabilities in each test sample. To validate our observations, we apply a sign test. The figure confirms the observation from before that the interval model with $n = 2$ intervals clearly performs best, having a by seven to eight percent higher area-under-curve than the next best model (p-value $\leq 10^{-44}$) for any number of rounds. It is followed by the time-dependent Markov model, which performs slightly but still statistically significantly ($p \leq 10^{-4}$) better than the interval model with $n = 3$ intervals. All three models exhibit clearly better results ($p \leq 10^{-44}$) than our baseline, the time-independent Markov model. Apparently, the additional information of more interaction rounds consistently helps the models to make a little bit better predictions, but the gains are smaller than the differences between the models. Note that the interval model with $n = 2$ intervals, given test samples without any interaction rounds, even outperforms the time-dependent Markov model on test samples with four interaction rounds ($p = 4.3\%$) and the other models with five interaction rounds ($p \leq 10^{-3}\%$), which carry significantly more information.

8 Conclusion

We concentrate on a scheduling setting in which the scheduler interacts with human users to create a suitable schedule and use Bayesian Learning to learn user behavior. In particular, we focus on the users' availability throughout the day and the resulting responses to the schedulers queries. Training data is sparse and collected from user interaction of previous problem instances. We propose and train three different models, two of them based on a two-state Markov chain with time-independent, resp. time-dependent transition probabilities, and the third model learns intervals in which the user is available. We compare the models on two different datasets that are based on artificially generated availabilities and the Dutch time-use-survey, resp. Results show that the time-dependent Markov model performs best and similar to the true model for the generated dataset, and the interval model with two intervals performs best for the other dataset. If the models are given less information from user interaction, they perform marginally worse, but these differences are smaller than the differences between the models.

So far, we evaluate the user models based on precision and recall. Future work should investigate the impact of the learned user model on the obtained schedules. Since the user model has influence on how training data for future iterations is collected, active learning appears to be a promising way to improve future predictions. As we use Bayesian Learning, our learned models provide a variance along with the acceptance probability that indicates whether training data or the data given with the sample is insufficient to make a definite prediction and gives a useful hint about the information gain of a query.

References

1. Anghinolfi, D., Paolucci, M., Ronco, R.: A bi-objective heuristic approach for green identical parallel machine scheduling. European Journal of Operational Research **289**(2), 416–434 (2021)
2. Blum, A.: Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Machine Learning **26**(1), 5–23 (1997)
3. Bolt, D.M., Piper, M.E., Theobald, W.E., Baker, T.B.: Why two smoking cessation agents work better than one: role of craving suppression. Journal of Consulting and Clinical Psychology **80**(1), 54–65 (2012)
4. Cheng, J., Chu, F., Zhou, M.: An improved model for parallel machine scheduling under time-of-use electricity price. IEEE Transactions on Automation Science and Engineering **15**(2), 896–899 (2018)
5. Chib, S.: Markov chain Monte Carlo methods: computation and inference. In: Heckman, J.J., Leamer, E. (eds.) Handbook of Econometrics, vol. 5, pp. 3569–3649. Elsevier (2001)
6. Ding, J.Y., Song, S., Zhang, R., Chiong, R., Wu, C.: Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. IEEE Transactions on Automation Science and Engineering **13**(2), 1138–1154 (2016)
7. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian data analysis. Chapman and Hall/CRC, New York (1995)

8. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence **6**(6), 721–741 (1984)
9. Hastings, W.K.: Monte carlo sampling methods using markov chains and their applications. Biometrika **57**(1), 97–109 (1970)
10. Horvitz, E.: Coordinate: Probabilistic forecasting of presence and availability. In: Proc. 18th Conference on Uncertainty and Artificial Intelligence (UAI02). pp. 224–233. Morgan Kaufmann Publishers (2002)
11. Horvitz, E., Apacible, J.: Learning and reasoning about interruption. In: Proceedings of the 5th International Conference on Multimodal Interfaces. pp. 20–27. Association for Computing Machinery (2003)
12. Horvitz, E., Koch, P., Apacible, J.: Busybody: creating and fielding personalized models of the cost of interruption. In: Proceedings of the 2004 ACM conference on Computer supported cooperative work. pp. 507–510. Association for Computing Machinery (2004)
13. Isaac, N.J., Jarzyna, M.A., Keil, P., Dambly, L.I., Boersch-Supan, P.H., Browning, E., Freeman, S.N., Golding, N., Guillera-Arroita, G., Henrys, P.A., et al.: Data integration for large-scale models of species distributions. Trends in Ecology & Evolution **35**(1), 56–67 (2020)
14. Kapoor, A., Horvitz, E.: Principles of lifelong learning for predictive user modeling. In: User Modeling 2007: 11th International Conference, UM 2007, Corfu, Greece, July 25-29, 2007. Proceedings 11. pp. 37–46. Springer (2007)
15. van de Meent, J.W., Paige, B., Yang, H., Wood, F.: An introduction to probabilistic programming. Preprint arXiv:1809.10756 (2018)
16. Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J., Zabowski, D., et al.: Experience with a learning personal assistant. Communications of the ACM **37**(7), 80–91 (1994)
17. Saberi-Aliabad, H., Reisi-Nafchi, M., Moslehi, G.: Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. Journal of Cleaner Production **249**, 119393 (2020)
18. van de Schoot, R., Depaoli, S., King, R., Kramer, B., Märtens, K., Tadesse, M.G., Vannucci, M., Gelman, A., Veen, D., Willemsen, J., et al.: Bayesian statistics and modelling. Nature Reviews Methods Primers **1**(1), 1 (2021)
19. Sociaal en Cultureel Planbureau: Tijdsbestedingsonderzoek 2005 - TBO 2005 (2005). `https://doi.org/10.17026/dans-znn-5xvz`
20. Varga, J., Raidl, G.R., Rönnberg, E., Rodemann, T.: Interactive job scheduling with partially known personnel availabilities. In: Dorronsoro, B., Chicano, F., Danoy, G., Talbi, E.G. (eds.) OLA 2023: Optimization and Learning. Communications in Computer and Information Science, vol. 1824, pp. 236–247. Springer (2023)
21. Varga, J., Raidl, G.R., Rönnberg, E., Rodemann, T.: Scheduling jobs using queries to interactively learn human availability times. Computers & Operations Research **167**, 106648 (2024)
22. Wang, S., Wang, X., Yu, J., Ma, S., Liu, M.: Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. Journal of Cleaner Production **193**, 424–440 (2018)