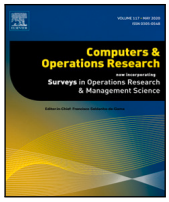
Computers & Operations Research 167 (2024) 106648

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



# Scheduling jobs using queries to interactively learn human availability times

Johannes Varga [a,*], Günther R. Raidl [a], Elina Rönnberg [b], Tobias Rodemann [c]

[a] Institute of Logic and Computation, TU Wien, 1040, Vienna, Austria
[b] Department of Mathematics, Linköping University, 581 83, Linköping, Sweden
[c] Honda Research Institute Europe, 63073, Offenbach, Germany

ARTICLE INFO

*Keywords:*
Job scheduling
Human machine interaction
Integer linear programming
Active learning

ABSTRACT

The solution to a job scheduling problem that involves humans as well some other shared resource has to consider the humans' availability times. For practical acceptance of a scheduling tool, it is crucial that the interaction with the humans is kept simple and to a minimum. It is rarely practical to ask users to fully specify their availability times or to let them enumerate all possible starting times for their jobs. In the scenario we are considering, users initially only propose a single starting time for each of their jobs and a feasible and optimized schedule shall then be found within a small number of interaction rounds. In each such interaction round, our scheduling approach may propose each user a small number of alternative time intervals for scheduling the user's jobs, and then the user may accept or reject these. To make the best out of these limited interaction possibilities, we propose an approach that utilizes integer linear programming and an exact and computationally efficient probability calculation for the users' availabilities assuming two different stochastic models. In this way, educated proposals of alternative time intervals for performing the jobs are determined based on the computed availability probabilities and the improvements these time intervals would enable. The approach is experimentally evaluated on a variety of artificial benchmark scenarios, and different variants are compared with each other and to diverse baselines. Results show that an initial schedule can usually be quickly improved over few interaction rounds even when assuming a quite simple stochastic model, and the final schedule may come close to the solution of the full-knowledge case despite the strongly limited interaction.

## 1. Introduction

We consider a class of job scheduling problems in which human users, e.g., the personnel of a company, is involved as a bottleneck resource. Jobs of these users have to be scheduled interactively in a way that is perceived by the humans as simple, stress-free, and with low cognitive effort, while at the same time a cost function has to be minimized. In the simplest, and from the users' perspective most convenient case, each user just suggests one starting time for each of their jobs. However, the jobs also require further shared resources, therefore this directly obtained schedule will rarely be feasible or cost-efficient. Ideally, we would know all about the times in which each user is available to perform their jobs. In this case we would be able to directly and optimally solve the scheduling problem without any further interaction. Requesting such complete availability information from the users is, however, in most practical scenarios impossible or far too troublesome. We therefore start with single starting time suggestions for the jobs from the respective users and collect more information on the users' availabilities in a small number of simple *interaction rounds*, which enhance the flexibility for the scheduling and help to find better schedules. In each such round, the scheduling approach is allowed to propose each user a small number of alternative time intervals for scheduling their jobs. The users are then supposed to indicate their acceptance or rejection of these time intervals in dependence of their availabilities. Hereby, we explicitly avoid that users need to specify larger amounts of additional availability intervals on their own. The insights gathered are used to improve the schedule from the previous round.

The main challenge we address in this work is to come up in each interaction round with *meaningful queries* for further time intervals to perform jobs in. These queries are presented to the users and should

(a) have a reasonable chance to be accepted and
(b) allow the optimization to obtain a better schedule.

* Corresponding author.
*E-mail addresses:* jvarga@ac.tuwien.ac.at (J. Varga), raidl@ac.tuwien.ac.at (G.R. Raidl), elina.ronnberg@liu.se (E. Rönnberg), tobias.rodemann@honda-ri.de (T. Rodemann).

https://doi.org/10.1016/j.cor.2024.106648
Received 4 September 2023; Received in revised form 3 April 2024; Accepted 4 April 2024
Available online 6 April 2024

Very large time intervals, for example, may improve the schedule the most due to the large gain of flexibility, but they will most likely be rejected by the users and cannot be considered meaningful. To consider (a), the likelihood that users accept queried time intervals in some reasonable way, we need to exploit at least some stochastic assumptions on the users' unknown availabilities. Ideally, we would have detailed user-specific stochastic models available, for example derived from historic availability data. Here, we first assume that such information is not known and instead build upon just a simple stochastic model, where the probability that the user is available in a timestep only depends on the availability in the previous timestep. For comparison purposes, we then also consider a more advanced stochastic model, which actually reflects the way our benchmark scenarios are generated.

The scheduling problem introduced in this paper will be called the *Interactive Job Scheduling Problem* (IJSP). It is inspired by an industrial setting where a company owns a few expensive motor test stands and employees use them to measure different characteristics of motors. We anticipate that this problem setting can be generalized to a class of problems where availability of employees and some resource of time-varying cost is needed. In the IJSP we assume that each job is associated with and requires one specific user and one of a set of available machines. On each machine, only one job can be performed at any time in a non-preemptive manner. As planning horizon we consider several days and time is discretized. Jobs have individual but machine-independent durations. Scheduling a job induces costs, for example for used electricity, and we consider these costs to be time-dependent. For example, when electricity is needed in a substantial way and bought on the spot market, (expected) electricity costs may change significantly over time. To avoid having to deal with infeasible schedules, we allow that jobs remain unscheduled at additional penalty costs. The objective is to find a feasible schedule of minimum total cost.

The overall scenario can also be seen as an *online* optimization problem, because important instance data only is revealed over time, but also as a kind of *active learning*, as the solution approach queries the users to learn more information, which is further exploited in the optimization. Our main contributions are

(a) to propose this general interactive scheduling setting and to narrow it down specifically to the IJSP for making concrete computational investigations on,
(b) an exact and computationally efficient calculation of the probabilities for users to accept potentially queried time intervals based on the already known availability information from the users and the assumption that the availability probability in a timestep only depends on the availability of the previous timestep,
(c) a respective probability calculation for a more advanced stochastic model reflecting the way our benchmark scenarios are created, and
(d) to propose a heuristic solution approach for the IJSP that utilizes these probability calculations.

In an experimental evaluation, this solution approach is compared to a greedy baseline approach as well as to solving the full-knowledge case. Results show that already with a very moderate amount of interaction and the assumptions of the simpler model, much better schedules can be obtained than with just the original user input. Moreover, the probability-based selection of time intervals to propose the users is clearly superior to the greedy method, and ultimately, schedules may be obtained that come close to those of solving the full-knowledge case.

A preliminary version of this work has been published as conference paper (Varga et al., 2023b). The current article extends this in several ways. Entirely new is the consideration of the advanced stochastic model, which assumes that a user is available in up to two time intervals per day, both with known distributions for their start time and duration. The comparison of the two different methods for calculating probabilities within our approach shows that the more advanced model leads to better results, although the simpler model still performs surprisingly well and is more generally applicable. Moreover, we considerably extended the experimental evaluation. In particular we now also

(a) vary the number of users while at the same time fixing the number of jobs,
(b) examine the convergence behavior of our approaches over a higher number of interaction rounds,
(c) consider different user-communication configurations by varying the amount of interaction done each round and requiring users to initially specify additional alternative starting times for each of their jobs, and
(d) evaluate on instances, where the user availabilities are derived from real-world data.

The next section puts our work in relation to other work from the literature. Section 3 formalizes our IJSP and introduces the ILP used as optimization core. Section 4 presents our solution approaches: a greedy baseline method and the advanced heuristic that makes use of estimated acceptance probabilities for time interval suggestions. The calculation of acceptance probabilities is subsequently detailed in Section 5 based on the simpler model and in Section 6 based on the more advanced model. Section 7 shows experimental results, and Section 8 concludes this work.

## 2. Related work

The core of this problem, if neglecting the users, can be described in the common three-field notation for scheduling problems (Graham et al., 1979) as Q||TEC. The objective is to minimize Total Energy Costs (TEC). Usually, uniform machines, which are indicated by Q, differ in their speed and therefore job processing times between any two machines differ by a constant factor. In our problem, job processing times are machine independent, still energy costs are uniformly machine dependent and thus we consider the machines to be uniform. The similar problem Q||$C_{\max}$, TEC, again with machine independent processing times and machine dependent energy costs, additionally takes the makespan into account for the objective and has been considered in the literature. Solving approaches for it include applying a solver to a Mixed Integer Linear Program (MILP) (Wang et al., 2018; Anghinolfi et al., 2021), a problem specific heuristic and a genetic algorithm (Wang et al., 2018), as well as a greedy heuristic and local search (Anghinolfi et al., 2021). Also similar is the scheduling problem R||TEC where jobs have in general different processing times on different machines. For this problem, Ding et al. (2016) proposed a MILP and a Dantzig–Wolfe decomposition. The MILP formulation was further improved by Cheng et al. (2018) and by Saberi-Aliabad et al. (2020).

The overall scenario can be seen as a Markov-decision process. User availabilities are only partially known and revealed over time based on the algorithm's actions. The algorithm's current knowledge about user availabilities is the state, the query is the action and the cost reduction after user replies is the reward. Markov-decision processes serve as underlying environment in reinforcement learning, which has been used in numerous works to solve scheduling problems. Kayhan and Yildiz (2021) review 80 papers on solving scheduling problems via reinforcement learning, published between 1995 and 2020, and identify characteristics such as the learning algorithm, the problem type and whether the system is modeled as single- or as multi-agent. Zhang et al. (2012) used reinforcement learning to solve another parallel machine scheduling problem. More recently, Monaci et al. (2024) solved the job shop scheduling problem using deep reinforcement learning, Uzunoglu et al. (2023) incorporated supervised learning to solve a scheduling problem with batching, and Varga et al. (2023a) introduced a Benders decomposition framework for scheduling problems in which the addition of cuts is effectively guided by a machine learning model. Note that

these approaches apply supervised learning and reinforcement learning to learn a model that is part of the optimization algorithm and thus they cannot be used for our scenario.

In interactive optimization approaches, most works only consider a single user who guides the optimization process. For instance, Saha et al. (2021) develop approaches based on evolutionary algorithms that cooperate with human designers to find aesthetic, aerodynamic, and structurally efficient designs for automotives. Furthermore, Aghaei-Pour et al. (2022) consider a multiobjective optimization problem where the human interactively specifies preferences on the solution, and those preferences are considered within an evolutionary algorithm. Interactive optimization with multiple users is less common. For instance, Jatschka et al. (2021) consider a MILP-based cooperative optimization approach that interacts with many users to learn an objective function for distributing service points in mobility applications. As optimization core they solve a MILP. In contrast to our problem setting they do not distinguish between different users and only learn about the collective preferences of all users.

Active learning on the availability times of the users has already been done in the domain of calendar scheduling. There, a calendar scheduling agent assists the user in arranging meetings with others and to do so it learns the user's preferences over time. Existing approaches use decision trees (Mitchell et al., 1994), the weighted-majority algorithm or the Winnow algorithm (Blum, 1997) for the learning task. In particular they learn to suggest a duration, location, day-of-the-week and time for a meeting, given the event type and the attendees. Since these approaches fundamentally rely on the event type and attendees, and do not consider a cost function, they cannot be applied in our setting without substantial modifications.

## 3. Interactive job scheduling problem

The IJSP is formally introduced as follows. Let the planning period be given by $t^{\text{max-day}}$ days, each with $t^{\text{max}}$ uniform timesteps, and let $T = \{t \mid t = (t^{\text{day}}, t^{\text{time}}),\ t^{\text{day}} = 1, \dots, t^{\text{max-day}},\ t^{\text{time}} = 1, \dots, t^{\text{max}}\}$ be a set of pairs where each pair refers to a specific timestep at a specific day. To refer to a time interval within a day and the corresponding set of timesteps, we use the notation $[t_1, t_2] = \{(t_1^{\text{day}}, t_1^{\text{time}}), \dots, (t_2^{\text{day}}, t_2^{\text{time}})\}$ for $t_1, t_2 \in T \mid t_1^{\text{day}} = t_2^{\text{day}},\ t_1^{\text{time}} \le t_2^{\text{time}}$, and adding a scalar $\Delta$ to a tuple $t \in T$ is defined as $t + \Delta = (t^{\text{day}}, t^{\text{time}} + \Delta)$.

Denote the set of users by $U$ and let the set of jobs of user $u \in U$ be $J_u$. Let each job $j \in J_u$ have a duration $d_j \in \{1, \dots, t^{\text{max}}\}$ and use the notation $T_j[t] = [t, t + d_j - 1]$ to refer to the subset of timesteps where job $j$ is performed if started at timestep $t$. Furthermore, the candidate starting times of job $j \in J$ are restricted to the set $T_j^{\text{job}} = \bigcup_{t^{\text{day}}=1}^{t^{\text{max-day}}} \{(t^{\text{day}}, 1), \dots, (t^{\text{day}}, t^{\text{max}} - d_j + 1)\}$, since a job has to finish on the same day it started. Denote the set of all jobs by $J = \bigcup_{u \in U} J_u$, and let $n = |J|$. To perform a job, two resources are needed: the user associated with the job and a machine. Denote the set of machines by $M$.

Using machine $i \in M$ in timestep $t \in T$ induces a time-dependent cost $c_{it} \ge 0$, e.g., for electricity depending on expected spot market prices. For a job to be feasibly scheduled, it needs to be given non-preemptive access to a machine and its user must have time for the complete duration of the job. If a job $j \in J$ cannot be feasibly scheduled, this induces a cost $q_j \ge 0$, e.g., for over-time or extra personnel. We assume that the cost for leaving a job unscheduled is always higher than the highest cost of scheduling it, i.e., $q_j \ge d_j \max_{i \in M, t \in T} c_{it}$, $j \in J$.

The dynamic and interactive aspect of our problem is represented by $\mathcal{T} = (\mathcal{T}_j)_{j \in J}$ where $\mathcal{T}_j \subseteq T_j^{\text{job}}$ are the timesteps in which job $j$ may start in when considering the respective user's currently known availabilities. More details on $\mathcal{T}$ are addressed later.

Assuming for now $\mathcal{T}$ is given and fixed, we aim at finding a feasible schedule of minimum cost. The problem is strongly NP-hard and cannot be approximated, as it is a generalization of the basic STOUC, introduced by Chen and Zhang (2019), which is also NP-hard and cannot be approximated. We model the IJSP with the following ILP, in which the binary decision variables $x_{jit}$ indicate if job $j \in J$ is scheduled on machine $i \in M$ to start with timestep $t \in \mathcal{T}_j$, or not.

$$\text{ILP}(\mathcal{T}) \quad \min \quad \sum_{j \in J} \sum_{i \in M} \sum_{t \in \mathcal{T}_j} \sum_{t' \in T_j[t]} c_{it'} x_{jit} + \sum_{j \in J} q_j \left(1 - \sum_{i \in M} \sum_{t \in \mathcal{T}_j} x_{jit}\right) \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in M} \sum_{t \in \mathcal{T}_j} x_{jit} \le 1 \qquad j \in J \tag{2}$$

$$\sum_{j \in J} \sum_{t \in \mathcal{T}_j \mid t' \in T_j[t]} x_{jit} \le 1 \qquad i \in M,\ t' \in T \tag{3}$$

$$\sum_{j \in J_u} \sum_{i \in M} \sum_{t \in \mathcal{T}_j \mid t' \in T_j[t]} x_{jit} \le 1 \qquad u \in U,\ t' \in T \tag{4}$$

$$x_{jit} \in \{0, 1\} \qquad j \in J,\ i \in M,\ t \in \mathcal{T}_j \tag{5}$$

The first and second term of the objective function (1) correspond to the total cost for machine usage and unscheduled jobs, respectively. Constraints (2) ensure that each job is scheduled at most once, constraints (3) limit the number of scheduled jobs per machine and timestep to one, and constraints (4) limit the number of jobs per user and timestep to one.

As indicated, this model can be solved for different sets $\mathcal{T}$ that reflect the user availability information in the current stage of the decision-making. As an important characteristic of the problem is that the user availability is not assumed to be fully known, we introduce the following notation for the currently available information. Let $T_u^{\text{avail}} \subseteq T$ be a subset of timesteps where user $u \in U$ has confirmed to be available. Feasible start times for each job $j \in J_u$ can then be derived as $T_j^{\text{feas}} = \{t \in T_j^{\text{job}} \mid T_j[t] \subseteq T_u^{\text{avail}}\}$. Further, let $T_j^{\text{infeas}} \subseteq T$ refer to time steps where job $j \in J$ is not allowed to start since the user is known to be unavailable in at least one time step in $T_j[t]$, $t \in T_j^{\text{infeas}}$.

Based on these confirmed availabilities and unavailabilities, it is possible to solve the model ILP($\mathcal{T}$) for two extreme cases. For $\mathcal{T} = (T_j^{\text{feas}})_{j \in J}$, only the timesteps that the respective users have so far confirmed to be available are included, and thus the solution to ILP$((T_j^{\text{feas}})_{j \in J})$ is feasible for the IJSP and in general provides a pessimistic bound. For $\mathcal{T} = (T_j^{\text{job}} \setminus T_j^{\text{infeas}})_{j \in J}$, all timesteps except those where the users are already known to be not available are included, and the solution to ILP$((T_j^{\text{job}} \setminus T_j^{\text{infeas}})_{j \in J})$ provides an optimistic bound; but the corresponding schedule may not be feasible with respect to user availability.

The interactive aspect of the problem is that users can be queried concerning their availabilities. A query is represented by a pair $(u, [t, t'])$ specifying a user $u \in U$ and a time interval from $t \in T$ to $t' \in T$. If the user is available in the full interval of the query, this information is directly included in the sets $T_u^{\text{avail}}$ and $T_j^{\text{feas}}$. If the user is unavailable in at least one timestep of the interval, the interval is rejected and included in the set $I_u^{\text{rej}}$. In such update, $I_u^{\text{rej}}$ is made sure not to contain any interval that is a superinterval of another interval, as such superintervals are redundant. The interaction with the users is made in a number of rounds, and before each new round an updated ILP($\mathcal{T}$) can be solved. Let the number of rounds be $B \in \mathbb{N}_{>0}$, and let the allowed number of queries in each round be $b \in \mathbb{N}_{>0}$. In each round, one user may receive multiple queries, i.e., suggestions for alternative starting times for their jobs. The choice of queries to make in a round is critical for the outcome of the scheduling, and our strategy for this is described in the next section.

## 4. Solving approaches

The challenge in each round is to find a set of queries that are likely to be accepted and reduce the objective value as much as possible if accepted. We consider only queries that are reasonable in the following sense. They concern the scheduling of jobs outside the users' already known availabilities, and we do not want to have more than one query for a user for the same day. Denote with $T_j^{\text{query}} = T_j^{\text{job}} \setminus T_j^{\text{infeas}} \setminus T_j^{\text{feas}}$ all

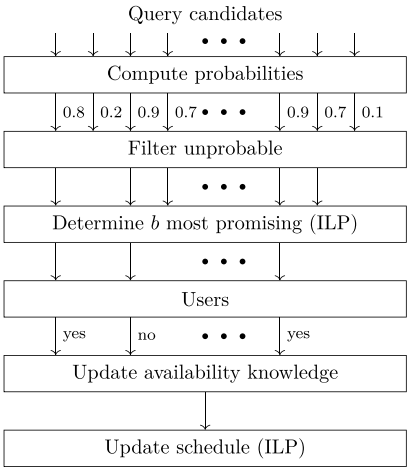



**Fig. 1.** Visualization of the solution approach for a single round.

starting times of job $j$ that would require a confirmed user query. Most beneficial queries – if accepted – can then be determined by solving the model ILP$((T_j^{\text{query}} \cup T_j^{\text{feas}})_{j\in J})$ with the additional constraints

$$\sum_{j\in J}\sum_{i\in M}\sum_{t\in T_j^{\text{query}}} x_{jit} \le b \tag{6}$$

$$\sum_{j\in J_u}\sum_{i\in M}\sum_{\tilde{t}\in T_j^{\text{query}}|\tilde{t}^{\text{day}}=t^{\text{day}}} x_{ji\tilde{t}} \le 1 \qquad u\in U, t^{\text{day}}\in\{1,\dots,t^{\text{max-day}}\} \tag{7}$$

where the former limits the total number of user queries to $b$ and the latter prevents multiple queries for the same user on the same day. Having obtained a solution $x$, each value of one of a variable $x_{jit}$ for $u \in U$, $j \in J_u$, $i \in M$ and $t \in T_j^{\text{job}} \setminus T_j^{\text{infeas}} \setminus T_j^{\text{feas}}$ results in a query $[t, t+d_j-1]$ for user $u$. We refer to this approach to determine user queries by Greedy. In particular, Greedy selects those queries that improve the objective value the most, if accepted.

This approach can possibly be improved by assuming that the user availabilities behave according to some model that yields an acceptance probability for each query. To exploit such probabilities, we remove the starting times from $T_j^{\text{query}}$ whose associated queries have probabilities below a given threshold $0 \le p^{\text{lim}} \le 1$, i.e., which we do not consider promising. Queries are again obtained by solving ILP$((T_j^{\text{query}} \cup T_j^{\text{feas}})_{j\in J})$ with constraints (6) and (7), but now with these reduced sets $T_j^{\text{query}}$. As model for the acceptance probabilities, the next section proposes a rather generic one based on the assumption that the availability probability of a user only depends on the availability in the previous timestep. This can also be seen as a two-state Markov process, and consequently, we refer to this model-based solution approach by Markov($p^{\text{lim}}$). Later, in Section 6, we will further consider a more specialized model for the probability calculation, which corresponds to the actual way our benchmark scenarios were randomly generated. We will refer to this as Advanced($p^{\text{lim}}$).

The whole procedure in each round is shown in Fig. 1. First the set of possible queries is determined and for each of these query candidates the acceptance probability of the user is estimated. Only queries with estimated probability above the threshold $p^{\text{lim}}$ are considered further and an ILP is used to select the $b$ most promising ones, in the sense that they improve the objective value most if accepted. The corresponding user either accepts or rejects each of these queries and this feedback is used to update the knowledge about the users' availabilities and to compute an improved schedule based on the gained freedom.

## 5. Probability calculation for the two-state Markov process

Consider a single user $u \in U$ and a single day $t^{\text{day}} \in \{1,\dots,t^{\text{max-day}}\}$. For better readability we refer to the timesteps of this day in the following by $T_{t^{\text{day}}} = \{1,\dots,t^{\text{max}}\}$. Assume that the average duration of the periods when a user is available, and the average duration of the unavailable-periods are known. When we want to exploit just this minimal information, it is natural to model a user's availabilities by a simple two-state Markov process. The two states of this process are 0 and 1, representing that the user either is unavailable in the current timestep or available, respectively. Moreover, let us introduce the additional artificial timesteps 0 and $t^{\text{max}}+1$ before the start of the day and after the end of the day. In both of these timesteps, the user is not available and therefore the corresponding state is 0. Proceeding from one timestep to the next, we associate probabilities $\rho_{00}$, $\rho_{01}$, $\rho_{10}$, and $\rho_{11}$ for staying in state 0, transitioning from 0 to 1, transitioning from 1 to 0, and staying in state 1, respectively. Naturally, $\rho_{00} = 1-\rho_{01}$ and $\rho_{11} = 1-\rho_{10}$ must hold. This Markov process is depicted in Fig. 2a. The transition probabilities are computed based on the fact that the expected number of steps the Markov process stays in state 1 is $1/\rho_{10}$ and $1/\rho_{01}$ for state 0. In this section we only consider one user, and for the sake of simplicity we omit the index regarding this user.

Given the current set of known availability times $T^{\text{avail}}$ and the set of so far rejected time intervals $I^{\text{rej}}$, we now want to determine the probability that the user is available in some given time interval $[\tau,\tau']$, $1 \le \tau \le \tau' \le t^{\text{max}}$. For this purpose we unroll the Markov process into a state graph over all timesteps from 0 to $t^{\text{max}}+1$ as follows and illustrated in Fig. 2b.

As the user is supposed to be not available outside of $T_{t^{\text{day}}}$, the initial state at the beginning of the day is represented by the single node $0_0$. Then, we have nodes $0_t$ and $1_t$ for each timestep $t \in T_{t^{\text{day}}}$, indicating the availability or non-availability of the user in timestep $t$. We also add node $0_{t^{\text{max}}+1}$ and for now $1_{t^{\text{max}}+1}$ to allow a correct modeling of the transition to the time after the considered time horizon by the two-state Markov process. All nodes of two successive timesteps are connected with arcs corresponding to the state transitions of the Markov process, and they are weighted with the respective transition probabilities $\rho_{00}$, $\rho_{01}$, $\rho_{10}$, and $\rho_{11}$.

Ignoring known user availabilities $T^{\text{avail}}$ and rejected time intervals $I^{\text{rej}}$ for now, this state graph has been constructed in such a way that each path from node $0_0$ to either node $0_{t^{\text{max}}+1}$ or $1_{t^{\text{max}}+1}$, which we call terminal nodes, corresponds to exactly one outcome of the Markov process over $t^{\text{max}}+1$ timesteps, and each possible outcome of the Markov process has an individual corresponding path. We refer by the *probability* of a path to the product of the path's arc weights, and with the probability of a set of paths to the sum of the paths' probabilities. The probability of all paths from node $0_0$ to any of the terminal nodes is then one as this covers all possible outcomes of the Markov process.

Next, we consider the already known availability times $T^{\text{avail}}$ of the user by removing all nodes $0_t$ for $T^{\text{avail}}$ with their incident arcs. This effectively reduces the set of possible paths, and thus represented Markov process outcomes, to those where state 1 is achieved in all timesteps from $T^{\text{avail}}$. Moreover, we also remove node $1_{t^{\text{max}}+1}$ with its ingoing arcs in order to model that the user is unavailable after the last actual timestep $t^{\text{max}}$.

To modify the graph w.r.t. the intervals in which the user is known to be available was straightforward since all timesteps of such intervals must have state 1. A time interval rejected by the user requires more care since it implies only that for *at least* one timestep in the interval – but not necessary all – the Markov process is in state 0. Only a rejected time interval $[t,t] \in I^{\text{rej}}$ of length one can thus be handled directly by removing node $1_t$ with its incident arcs as the Markov process has to be in state 0 in this timestep. For a longer rejected interval $[t_1,t_2] \in I^{\text{rej}}$ we ensure that only paths are kept in the graph where the Markov process achieves state 0 at least once within this interval. More specifically, observe that if the Markov process is in timestep $t \in [t_1,t_2]$ and state 0 has not been obtained in timesteps $[t_1,t]$ yet, then there has to follow at least one timestep $t' \in [t+1,t_2]$ in which state 0 is achieved. To model this aspect, we add further nodes $1_t^{t_2}$ for $t \in [t_1,t_2-1]$, $[t_1,t_2] \in I^{\text{rej}}$ to our graph. Former arcs $(0_t,1_{t+1})$ and $(1_t,1_{t+1})$, $t \in T_{t^{\text{day}}} \cup \{0\}$, are now

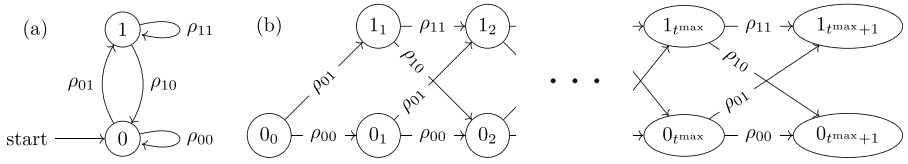



**Fig. 2.** (a) Two-state Markov process and (b) corresponding unrolled state graph.

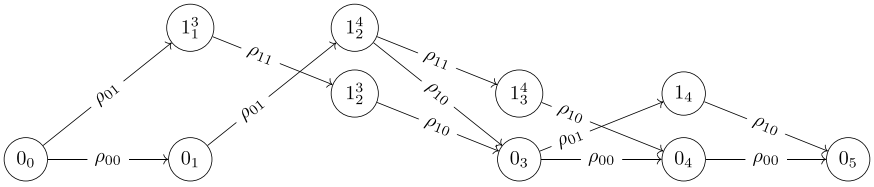


**Fig. 3.** The state graph for $t^{\max} = 4$, $T^{\text{avail}} = \{2\}$, and $I_u^{\text{rej}} = \{(1,3),(2,4)\}$.

replaced by arcs $(0_t, 1^{t_2}_{t+1})$ and $(1_t, 1^{t_2}_{t+1})$, respectively if there is a rejected time interval $[t_1, t_2] \in I^{\text{rej}}$ starting in the next timestep $t_1 = t+1$ and ending in timestep $t_2$. Note that there can be at most one interval in $[t_1, t_2] \in I^{\text{rej}}$ that starts at timestep $t_1$ since $I^{\text{rej}}$ has been guaranteed not to contain a proper subinterval of $[t_1, t_2]$. Each new node $1^{t_2}_t$ further has an outgoing arc to node $0_{t+1}$ if this node still exists, corresponding to the transition to state 0. Moreover, there is an outgoing arc from each node $1^{t_2}_t$ to node $1^{t_2}_{t+1}$ as long as $t+1 < t_2$ for the case of staying in state 1. Due to the absence of an arc from node $1^{t_2}_{t_2-1}$ to some successor node in which state 1 is kept, it is effectively enforced that state 0 is reached at least once within the rejected time interval $[t_1, t_2]$. Remaining nodes without ingoing arcs except $0_0$ and their outgoing arcs are pruned as they do not play an active further role. An example of such a final state graph is shown in Fig. 3.

Now, we want to utilize this graph to derive the probability that the considered user is available in a given time interval $[\tau, \tau']$. The key observation to do this efficiently is that each path from node $0_0$ to a node $v$ passes through exactly one predecessor of $v$. Therefore the total probability $p^{\text{path}}_{0_0,v}$ of all paths from $0_0$ to $v$, denoted by Paths$(0_0, v)$, can be computed recursively as

$$p^{\text{path}}_{0_0,v} = \sum_{P \in \text{Paths}(0_0,v)} \prod_{(u,u') \in P} \rho(u,u') = \sum_{u \in N^-(v)} \sum_{P \in \text{Paths}(0_0,u)} \left( \prod_{(u,u') \in P} \rho(u,u') \right) \cdot \rho(u,v) = \sum_{u \in N^-(v)} p^{\text{path}}_{0_0,u} \rho(u,v), \tag{8}$$

where $P$ denotes one specific $0_0$–$v$ path represented by the corresponding set of arcs and $N^-(v)$ is the set of predecessors of node $v$. Denoting the set of successors of node $v$ by $N^+(v)$, the probabilities $p^{\text{path}}_{v,0_{t^{\max}+1}}$ of all paths from a node $v$ to node $0_{t^{\max}+1}$ can be correspondingly computed recursively by

$$p^{\text{path}}_{v,0_{t^{\max}+1}} = \sum_{w \in N^+(v)} p^{\text{path}}_{w,0_{t^{\max}+1}} \rho(v,w). \tag{9}$$

We are now interested in all those paths from $0_0$ to $0_{t^{\max}+1}$ that stay for the timesteps $\tau$ to $\tau'$ in state 1 nodes, indicating the availability of the user. Each of these paths is composed of a path from $0_0$ to $1^{t_2}_\tau$, a path $P$ from $1^{t_2}_\tau$ to $1^{t_2}_{\tau'}$ that only uses state 1 nodes, and a path from $1^{t_2}_{\tau'}$ to $0_{t^{\max}+1}$ for some $t_2 \geq \tau'+1$. As a special case the middle segment $P$ can also start in $1_\tau$ and then it either ends in $1_{\tau'}$ if no rejected interval starts within $[\tau, \tau']$ or otherwise in $1^{t_2}_{\tau'}$ for an appropriate $t_2 \geq \tau'+1$. There are only a few possibilities for the middle segment $P$ and the probability of all paths that stay in state 1 nodes for the timesteps from $\tau$ to $\tau'$ can be computed with a sum over these possibilities. For us, the conditional probability in respect to all paths in the graph, i.e., those respecting $T^{\text{avail}}$ and $I^{\text{rej}}$ and ending in $0_{t^{\max}+1}$, is of main interest, which is

$$p^{\text{avail}}([\tau,\tau'] \mid T^{\text{avail}}, I^{\text{rej}}, 0_{t^{\max}+1}) = \frac{\sum_{P \in \text{1-Paths}(\tau,\tau')} p^{\text{path}}_{0_0,P_\tau} \cdot \rho_{11}^{\tau'-\tau} \cdot p^{\text{path}}_{P_{\tau'},0_{t^{\max}+1}}}{p^{\text{path}}_{0,t^{\max}+1}}, \tag{10}$$

where the sum is taken over all middle segments 1-Paths$(\tau,\tau')$, and $P_\tau$ and $P_{\tau'}$ are the first and last nodes of a middle segment $P$, respectively. The denominator is the probability of all paths from $0_0$ to $0_{t^{\max}+1}$, and the nominator the probability of only those paths that stay in state 1 nodes in timesteps $\tau$ to $\tau'$.

*Time complexity.* The unrolled state graph, considering rejected time intervals, can have for each timestep $\tau$ a 0-state $0_\tau$, a 1-state $1_\tau$ and 1-states $1^{t_2}_\tau$ with $t_2$ being the final timestep of each rejected time interval containing timestep $\tau$, i.e., $t_2 \in \widehat{I^{\text{rej}}}_\tau = \{t_2 \in T \mid \exists [t_1,t_2] \in I^{\text{rej}},\ t_1 \leq \tau \leq t_2\}$. Let $n^{\text{rej,max}} = \max_{\tau \in T} |\widehat{I^{\text{rej}}}_\tau|$ be the maximum number of rejected time intervals containing the same time step $\tau$. As the timesteps range from 0 to $t^{\max}+1$, the graph has at most $(n^{\text{rej,max}}+2)(t^{\max}+2)$ nodes and since no node can have more than two outgoing arcs, the number of arcs is limited by $2(n^{\text{rej,max}}+2)(t^{\max}+2)$. Computing $p^{\text{path}}$ is performed $\mathcal{O}(|U| t^{\text{max-day}})$ times and its time complexity is linear in the number of nodes and arcs and therefore in $\mathcal{O}(n^{\text{rej,max}} t^{\max})$. The number of 1-Paths in (10) is limited by the number of nodes in timestep $\tau$, since each 1-node can only have one succeeding 1-node. Therefore computing $p^{\text{avail}}([\tau,\tau'] \mid T^{\text{avail}}, I^{\text{rej}}, 0_{t^{\max}+1})$ takes time $\mathcal{O}(n^{\text{rej,max}})$ and is performed for each user, each day and each way to fit an interval of job length into the time horizon, in other words for $\mathcal{O}(|U| t^{\text{max-day}} t^{\max} \max_{j \in J} d_j)$ queries in each round. This results in an overall time complexity of $\mathcal{O}(|U| t^{\text{max-day}} t^{\max} n^{\text{rej,max}} \max_{j \in J} d_j)$ for computing the probabilities of all queries in one round.

## 6. Probability calculation for the advanced model

The two-state Markov process is a quite general but crude simplification of the users' behavior. To evaluate the impact of this simplification and to demonstrate how the approach can be extended to a more realistic scenario, we now consider a more advanced model. In fact, the benchmark instances we will use in the computational experiments were also generated according to this model.

Users are assumed to have up to two independent availability intervals each day, each happening with probability $q \in [0,1]$. The starting time and duration of each interval $i \in \{1,2\}$ are assumed to be distributed according to some (discrete) distributions with probability mass functions $f_i^{\text{start}}$ and $f_i^{\text{dur}}$, and cumulative distribution functions $F_i^{\text{start}}$ and $F_i^{\text{dur}}$, respectively. For more details on the distribution functions we use in our experimental evaluation, we refer to Section 7.1.

We derive a Markov chain to model this behavior and use the same ideas as in the previous section to calculate probabilities for queries

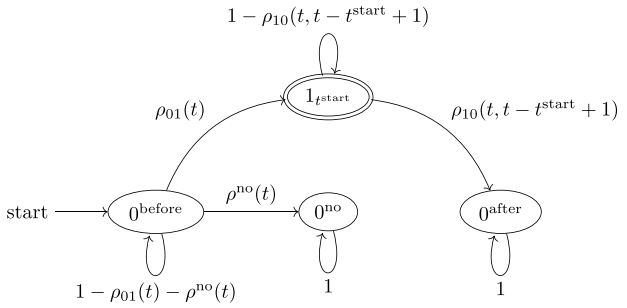



**Fig. 4.** Markov chain that models the random creation of an interval using given distributions for starting time and duration.

to be accepted. A single availability interval can be represented by the (time-dependent) Markov chain shown in Fig. 4. Here, $1_{t^{\text{start}}}$ represents multiple states, one for each timestep $t^{\text{start}}$ in which the interval may start. In timestep 0, the Markov chain starts in node $0^{\text{before}}$. It then transitions with probability $1-q$ to state $0^{\text{no}}$, corresponding to the case that there is no availability interval, and consequently no 1-node will be reached. This is represented by

$$\rho^{\text{no}}(t) = \begin{cases} 1-q & \text{if } t = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Else, each timestep $t$, $0^{\text{before}}$ transitions into state $1_{t+1}$ with probability

$$\rho_{01}(t) = \begin{cases} q \cdot f^{\text{start}}(1) & \text{if } t = 0 \\ \frac{f^{\text{start}}(t)}{1-F^{\text{start}}(t-1)} & \text{otherwise,} \end{cases} \tag{12}$$

which represents the beginning of the availability interval realizing the distribution $F^{\text{start}}$. Note that in the second case the probability is taken under the condition that the chain stayed in state $0^{\text{before}}$ until timestep $t$. From state $1_{t^{\text{start}}}$, the chain transitions into state $0^{\text{after}}$, which corresponds to ending the availability interval, with probabilities

$$\rho_{10}(t, t^{\text{dur}}) = \begin{cases} 1 & \text{if } t = t^{\max} \\ \frac{f^{\text{dur}}(t^{\text{dur}})}{1-F^{\text{dur}}(t^{\text{dur}}-1)} & \text{otherwise,} \end{cases} \tag{13}$$

where $t^{\text{dur}} = t - t^{\text{start}} + 1$.

Let $G_1 = (V_1, E_1, \rho_1)$ be the state graph for the first interval and $G_2 = (V_2, E_2, \rho_2)$ be the corresponding one for the second interval. The combined state graph, which represents the union of both intervals, is the direct product $G_1 \times G_2 = (V_1 \times V_2, E_{12}, \rho_{12})$ (Hammack et al., 2011, Chapter I.5), where each pair of arcs $(v_1, w_1) \in E_1$ and $(v_2, w_2) \in E_2$ results in one arc of $E_{12}$ with weight $\rho_1(v_1, w_1) \cdot \rho_2(v_2, w_2)$. The user is considered to be available in a node $(v_1, v_2) \in G_1 \times G_2$, or equivalently $(v_1, v_2)$ is a 1-node, iff at least one of $v_1$ and $v_2$ is a 1-node. All other nodes are 0-nodes.

The former knowledge about the user, $T^{\text{avail}}$ and $I^{\text{rej}}$, respectively, is considered in a similar way as in the previous section. First the product graph is unrolled, replicating each state in each timestep, with a single 0-node $0_0$ for timestep 0 and a single 0-node $0_{t^{\max}+1}$ for timestep $t^{\max}+1$. Then in the timesteps $t \in T^{\text{avail}}$, in which the user is available, all 0-nodes are removed. Intervals in $I^{\text{rej}}$ are taken into account in a similar way as discussed in the previous section. In particular, for each 1-node $v$ and each interval end $t_2 \in T : [t_1, t_2] \in I^{\text{rej}}$, we add a node $v^{t_2}$. Arcs $(u, v)$ with $u$ being a node of timestep $t$ are replaced with arcs $(u, v^{t_2})$, if $t_1 = t + 1$ for some interval $[t_1, t_2] \in I^{\text{rej}}$ that starts in $t_1$ and ends in $t_2$. For each outgoing arc from a node $v$ to a 0-node $w$ we add to each node $v^{t_2}$ an outgoing arc to node $w$. Similarly, we add an outgoing arc to node $w^{t_2}$ from each node $v^{t_2}$ for each outgoing arc from a node $v$ to a 1-node $w$, but only if $t < t_2 - 1$, where $t$ is the timestep of node $v$. Finally we recursively prune all nodes with no incoming or no outgoing arcs except for $0_0$ and $0_{t^{\max}+1}$.

To get the probability that the user accepts an interval $[\tau, \tau']$, we again use $p^{\text{path}}$ from (8) and (9), computed for the constructed graph. As 1-states can have multiple 1-state successors, we cannot use Eq. (10). Instead, for the nominator, we repeat the computation of $p^{\text{path}}_{0_0,v}$ from (8), but ignore 0-nodes in the interval $[\tau, \tau']$. The denominator is kept the same. To speed up the process, we start from the precomputed values $p^{\text{path}}_{0_0,v}$ for each node $v$ of timestep $\tau$, only make the computations up to the nodes of timestep $\tau'$, account for the remaining parts of the paths by multiplying the so far computed values with $p^{\text{path}}_{v',0_{t^{\max}+1}}$ for each node $v'$ of timestep $\tau'$, and finally take the sum over those products.

*Time complexity.* The state graph in Fig. 4 has up to $3 + t^{\max}$, and its product graph up to $(3 + t^{\max})^2$ nodes. To consider $I^{\text{rej}}$, at most $n^{\text{rej,max}} \cdot (3 + t^{\max})^2$ nodes are added for each timestep, where $n^{\text{rej,max}} = \max_{\tau \in T} |\{t_2 \in T \mid \exists [t_1, t_2] \in I^{\text{rej}},\ t_1 \le \tau \le t_2\}|$ is again the maximal number of overlapping rejected time intervals. Therefore the graph has $\mathcal{O}(n^{\text{rej,max}} \cdot (t^{\max})^3)$ nodes and computing $p^{\text{path}}$ has the same time complexity $\mathcal{O}(n^{\text{rej,max}} \cdot (t^{\max})^3)$, following the same argumentation as in the previous section. Computing the acceptance probability of a single query $[\tau, \tau']$ takes time $\mathcal{O}((\tau' - \tau) n^{\text{rej,max}})$ and therefore the overall time complexity for computing the probabilities of all queries in one round is $\mathcal{O}(|U| t^{\text{max-day}} (t^{\max})^3 n^{\text{rej,max}} (\max_{j \in J} d_j)^2)$.

## 7. Experimental evaluation

We implemented the approaches in Julia 1.9, using the MILP solver Gurobi 10.0 (https://www.gurobi.com) and the package JuMP as interface to Gurobi. As real world instances were not available to us we created artificial benchmark instances and used them to compare the approaches with each other. Each test run was performed on a single core of an AMD EPYC 7402 and Gurobi was given a timelimit of 15 min for each ILP, which led to final gaps below 1% for 99.85% of the ILPs and final gaps below 7% for the remaining 0.15%.

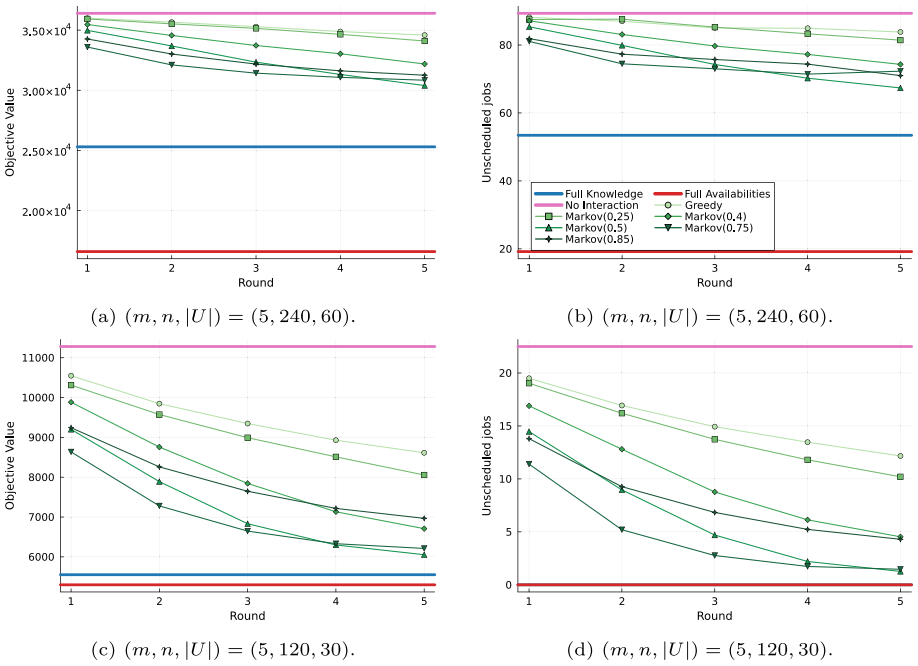
### 7.1. Instance generation

We consider a time horizon of $t^{\max} = 5$ days, each starting at 6 am and ending at 10 pm, with a time granularity of 15 min per timestep. Random time intervals are determined by a function rand_interval($\mu^{\text{start}}$, $\sigma^{\text{start}}$, $\mu^{\text{dur}}$, $\sigma^{\text{dur}}$) that first draws a random value from a normal distribution with mean $\mu^{\text{start}}$ and standard deviation $\sigma^{\text{start}}$ and rounds it to the closest timestep in $T$, which is then the start of the time interval. The duration of the interval is then determined by drawing another random value from a normal distribution with mean $\mu^{\text{dur}}$ and standard deviation $\sigma^{\text{dur}}$, rounding it to the closest positive integer. Should the interval exceed $t^{\max}$, it is capped at this last timestep of our time horizon. This generation process results in discrete probability mass functions $f^{\text{start}}$ and $f^{\text{dur}}$ by condensing the probability of the ranges that lead to the rounded and capped values into points, and those functions are used for the calculation described in Section 6.

Each user $u \in U$ is assumed to be available for a set of timesteps $T^{\text{avail}*}_u$. This set is determined independently for each day as follows. With a probability of 90%, the user is assumed to be available in rand_interval(9 am, 1 h, 4 h, 1 h) and, again with a probability of 90%, the user is assumed to be available in rand_interval(1 pm, 1 h, 5 h, 1 h). If the two intervals overlap the union is taken.

For each job $j \in J_u$ of user $u \in U$, the duration $d_j$ is chosen uniformly at random from 30 min to 4 h. Moreover, a starting time $t_j$ that allows the complete job to be scheduled within $T^{\text{avail}*}_u$ is selected uniformly at random. The initially provided set of availabilities for user $u \in U$ is then $T^{\text{avail}}_u = \bigcup_{j \in J_u} T_j[t_j]$.

We generated 30 instances for $m = 1, \ldots, 5$ machines with either 24 or 48 jobs per machine, i.e., $n \in \{24m, 48m\}$, and made them




(a) $(m, n, |U|) = (5, 240, 60)$. (b) $(m, n, |U|) = (5, 240, 60)$.

(c) $(m, n, |U|) = (5, 120, 30)$. (d) $(m, n, |U|) = (5, 120, 30)$.

**Fig. 5.** Development of the objective value (a, c) and the number of unscheduled jobs (b, d) when applying Greedy and Markov with different acceptance thresholds for two different instance sizes.

available online.[1] When considering the generated user availabilities, each machine can thus execute about 30 jobs on average, and for $n = 24m$ usually it is possible to schedule all jobs, while for $n = 48m$ this is not the case. We study different cases where each user has 2, 4 and 6 jobs. This results in either $4m$, $6m$, $8m$, $12m$, or $24m$ users for an instance. We allow $|U|$ user queries in each round, for a total of five rounds.

Costs for machine usage are based on real-world spot market prices $c_t^{\text{kWh}}$ for electricity in Germany for week 26 in 2022 taken from https://energy-charts.info. We use as cost $c_{i,t} = 15\text{min} \cdot P_i c_t^{\text{kWh}}$, where the electric power $P_i$ is assumed to differ among the machines $i \in M$ and is thus chosen uniformly at random from [50 kW,150 kW]. The cost $q_j$ for not scheduling a job $j \in J$ is set to 40 Euro $\cdot\, d_j$, which is roughly twice the cost of scheduling the job in the most expensive timesteps.

### 7.2. Comparison of the approaches

We performed simulations for Greedy as well as Markov($p^{\text{lim}}$) and Advanced($p^{\text{lim}}$) with acceptance probability thresholds $p^{\text{lim}} \in \{0.25, 0.4, 0.5, 0.75, 0.85\}$ on all benchmark instances. Note that with the advanced model, more queries $t^{\text{start}} \in T^{\text{infeas}}$ will in general be rejected. This knowledge can be used in Greedy and we will refer to this version with Greedy(Advanced). After each round we determine the best schedule that is feasible for the information collected up to this round. Fig. 5 shows the development of the mean objective value and the mean number of unscheduled jobs, respectively, over the rounds for Greedy and Markov on two different instance sizes. Values are aggregated over the 30 instances with $m = 5$ machines and $n = 120$ respectively $n = 240$ jobs. Furthermore, we determine the best feasible schedule with the information that is available before the first round ("No Interaction"), the best schedule when assuming that all users are available all the time ("Full Availabilities"), and the best schedule with full knowledge about the users' availabilities ("Full Knowledge") and show these as horizontal lines in the figures. Table 1 additionally shows the mean optimality gaps of the objective values from Greedy, Markov($p^{\text{lim}}$) and Advanced($p^{\text{lim}}$) in respect to "Full Knowledge" after five rounds in percent. Naturally, the objective value of the "Full Availabilities" scenario has to be less than (or equal to) the objective value of "Full Knowledge", which has to be less than (or equal to) the objective value of "No Interaction". The objective values of Greedy, Markov($p^{\text{lim}}$) and Advanced($p^{\text{lim}}$) have to be between the objective values of "Full Knowledge" and "No Interaction". Note that the plots 5(a) and 5(b) are also representative for the general trends with $(n, |U|) = (48m, 12m)$, and the same applies for plots 5(c) and 5(d) and $(n, |U|) = (24m, 6m)$.

We observe that Markov(0.5) and Markov(0.75) quickly converge towards the best possible schedule. For $n = 120$, the original objective values without interaction could almost be halved after five rounds, while for $n = 240$, 16% and 15% of the original costs could be saved after five rounds. Moreover, for $n = 120$, the final optimality gaps of these two approaches are by a factor of more than four better than the final gap of Greedy. In contrast, $p^{\text{lim}} = 0.25$ leads to much slower convergence with an improvement over Greedy of only roughly 17%. The objectives for $p^{\text{lim}} = 0.4$ and $p^{\text{lim}} = 0.85$ lie inbetween, being by a factor of more than two better than the final gap of Greedy. Remarkably, Markov(0.75) performs best in the first rounds, while Markov(0.5) catches up later on and performs best in the end. The reason is that the two-state Markov process has the steady state between 0.5 and 0.75 and therefore Markov(0.75) does not query days it knows nothing about while Markov(0.5) does; while it takes more iterations to get enough information about these days, this information provides more flexibility in scheduling the jobs. The table confirms our observations that Greedy consistently performs worse than Markov and that the values 0.5 and 0.75 are better suited for $p^{\text{lim}}$ than 0.25, 0.4 and 0.85.

Considering Advanced, Fig. 6 shows the objective value and the number of unscheduled jobs over the rounds, respectively, and Fig. 7 compares Markov to Advanced for $p^{\text{lim}} \in \{0.25, 0.5\}$. The objectives for Advanced($p^{\text{lim}}$) with different $p^{\text{lim}}$ are closer together than those for Markov, and thus the advanced model is less sensitive to the choice of $p^{\text{lim}}$. A value of $p^{\text{lim}} = 0.5$ performs best for instances with $(m, n, |U|) = (5, 120, 30)$ and Table 1 shows that values for $p^{\text{lim}}$ from $\{0.25, 0.4, 0.5\}$ perform best for other instance sizes. Unsurprisingly, the advanced model leads to better results with a gap that is down to a factor of 0.32 smaller after five rounds. However, we emphasize that Markov

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

**Table 1**
Mean %-gaps of the objective values after five interaction rounds for Markov($p^{\text{lim}}$) and Advanced($p^{\text{lim}}$) with different limits $p^{\text{lim}}$ and for Greedy with and without the advanced model. The median runtime per interaction round is given for each instance size, aggregated over methods and interaction rounds.

| $m$ | $n$ | $\lvert U\rvert$ | Markov | | | | | | Advanced | | | | | | Time [s] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Greedy | 0.25 | 0.4 | 0.5 | 0.75 | 0.85 | Greedy | 0.25 | 0.4 | 0.5 | 0.75 | 0.85 | |
| 1 | 24 | 12 | 48.7 | 39.5 | 26.3 | **21.8** | 31.0 | 63.2 | 49.5 | 21.7 | 15.1 | **10.2** | 13.9 | 53.7 | 8.9 |
| 1 | 24 | 6 | 75.9 | 39.0 | 33.2 | 34.1 | **31.9** | 65.8 | 69.9 | 18.5 | 18.3 | **12.2** | 21.2 | 43.9 | 5.4 |
| 1 | 24 | 4 | 97.6 | 68.0 | 45.3 | 42.0 | **41.9** | 70.0 | 97.3 | 32.6 | 30.3 | **20.9** | 23.5 | 46.9 | 4.0 |
| 1 | 48 | 24 | 47.1 | 43.2 | 33.8 | **28.8** | 36.2 | 41.5 | 44.3 | **27.2** | 27.8 | 30.2 | 40.4 | 51.2 | 17.0 |
| 1 | 48 | 12 | 37.8 | 32.1 | 25.6 | **19.7** | 25.7 | 27.1 | 37.0 | 17.0 | **15.9** | 19.7 | 26.8 | 32.9 | 10.1 |
| 1 | 48 | 8 | 44.0 | 38.2 | 28.1 | 22.4 | **21.4** | 24.7 | 44.1 | 19.2 | **16.0** | 17.1 | 23.9 | 31.3 | 7.6 |
| 2 | 48 | 24 | 53.3 | 42.6 | 19.4 | **13.9** | 16.2 | 36.1 | 48.7 | 10.1 | 5.5 | **4.5** | 10.9 | 26.1 | 19.2 |
| 2 | 48 | 12 | 77.3 | 50.6 | 26.0 | **13.2** | 14.6 | 38.8 | 70.9 | 9.8 | 5.8 | **4.9** | 8.4 | 23.1 | 22.1 |
| 2 | 48 | 8 | 94.3 | 68.3 | 39.0 | 30.7 | **23.1** | 47.8 | 88.5 | 24.3 | 16.8 | **11.6** | 15.8 | 29.6 | 20.5 |
| 2 | 96 | 48 | 43.0 | 41.8 | 32.6 | **27.3** | 33.6 | 37.5 | 43.5 | **25.5** | 26.9 | 29.8 | 41.1 | 49.8 | 43.0 |
| 2 | 96 | 24 | 36.7 | 32.7 | 24.5 | **18.3** | 20.8 | 23.7 | 36.2 | 16.0 | **15.9** | 18.7 | 27.1 | 33.2 | 25.0 |
| 2 | 96 | 16 | 39.8 | 36.5 | 27.4 | 20.1 | **19.0** | 21.4 | 40.0 | 15.2 | **14.2** | 16.2 | 24.2 | 30.3 | 18.0 |
| 3 | 72 | 36 | 37.4 | 34.2 | 15.4 | **9.8** | 10.6 | 25.3 | 38.0 | 6.0 | **4.4** | 4.8 | 9.3 | 20.0 | 40.2 |
| 3 | 72 | 18 | 71.9 | 52.9 | 32.3 | 14.9 | **13.6** | 32.0 | 68.2 | 11.6 | 7.8 | **6.9** | 10.3 | 22.7 | 177.2 |
| 3 | 72 | 12 | 87.4 | 59.9 | 41.2 | 22.8 | **14.4** | 35.2 | 87.5 | 14.8 | 8.8 | **7.5** | 9.1 | 20.9 | 286.9 |
| 3 | 144 | 72 | 44.9 | 41.6 | 32.6 | **27.5** | 34.1 | 38.4 | 44.4 | **25.9** | 27.5 | 31.0 | 43.2 | 52.5 | 116.4 |
| 3 | 144 | 36 | 37.1 | 35.2 | 26.8 | **19.4** | 21.8 | 24.0 | 37.5 | **17.0** | 17.1 | 20.1 | 28.9 | 35.4 | 51.4 |
| 3 | 144 | 24 | 36.8 | 34.4 | 26.6 | 20.5 | **18.6** | 20.0 | 37.2 | 14.9 | **14.4** | 16.0 | 24.1 | 30.4 | 38.8 |
| 4 | 96 | 48 | 32.9 | 22.6 | 12.5 | **9.2** | 11.5 | 22.0 | 30.2 | 4.9 | **4.8** | 5.3 | 11.1 | 18.5 | 83.5 |
| 4 | 96 | 24 | 65.5 | 53.3 | 23.6 | **11.5** | 12.2 | 27.8 | 67.7 | 9.9 | 5.9 | **5.4** | 8.5 | 16.4 | 901.7 |
| 4 | 96 | 16 | 79.1 | 65.3 | 37.2 | 19.1 | **9.9** | 29.4 | 80.4 | 17.2 | 10.9 | **6.9** | 8.1 | 15.0 | 909.8 |
| 4 | 192 | 96 | 40.3 | 37.6 | 29.5 | **25.4** | 31.3 | 34.7 | 39.9 | **23.6** | 25.7 | 28.5 | 39.8 | 48.2 | 196.0 |
| 4 | 192 | 48 | 35.5 | 35.0 | 26.6 | **19.3** | 21.7 | 23.5 | 35.4 | **16.3** | 16.7 | 19.3 | 28.8 | 35.2 | 101.5 |
| 4 | 192 | 32 | 37.5 | 34.4 | 26.9 | 20.6 | **19.0** | 20.5 | 37.3 | 15.3 | **14.8** | 16.9 | 24.7 | 30.5 | 75.1 |
| 5 | 120 | 60 | 22.2 | 20.4 | 10.1 | **8.8** | 10.2 | 17.4 | 22.6 | 4.7 | 5.0 | **4.6** | 10.2 | 18.1 | 168.8 |
| 5 | 120 | 30 | 56.3 | 46.5 | 21.4 | **9.1** | 11.8 | 25.5 | 60.1 | 7.2 | 5.8 | **4.6** | 9.3 | 16.6 | 914.0 |
| 5 | 120 | 20 | 80.3 | 58.8 | 34.6 | 18.9 | **12.6** | 30.4 | 81.6 | 15.2 | 9.2 | **7.0** | 8.2 | 16.9 | 917.7 |
| 5 | 240 | 120 | 42.4 | 39.7 | 31.7 | **27.5** | 33.3 | 36.7 | 42.1 | **25.3** | 27.2 | 31.0 | 42.7 | 51.9 | 326.4 |
| 5 | 240 | 60 | 37.3 | 35.3 | 27.7 | **20.5** | 22.3 | 23.9 | 37.5 | **16.9** | 17.6 | 20.3 | 29.4 | 36.7 | 206.7 |
| 5 | 240 | 40 | 34.4 | 31.4 | 24.3 | 18.4 | **17.3** | 18.7 | 34.4 | 13.7 | **13.6** | 15.4 | 23.4 | 29.1 | 141.4 |

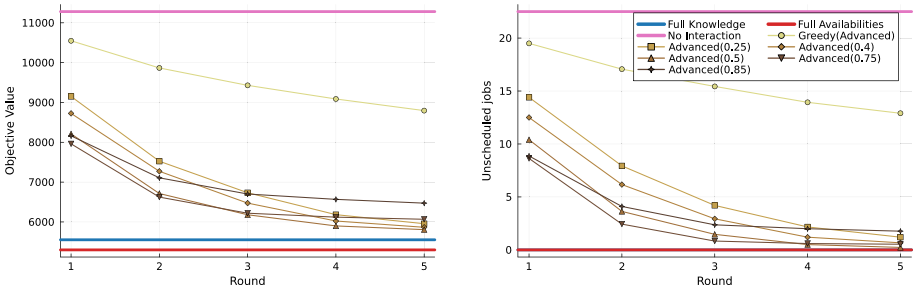



**Fig. 6.** Development of the objective value and the number of unscheduled jobs for instances with $(m, n, |U|) = (5, 120, 30)$ where the probabilities are calculated with the advanced model.

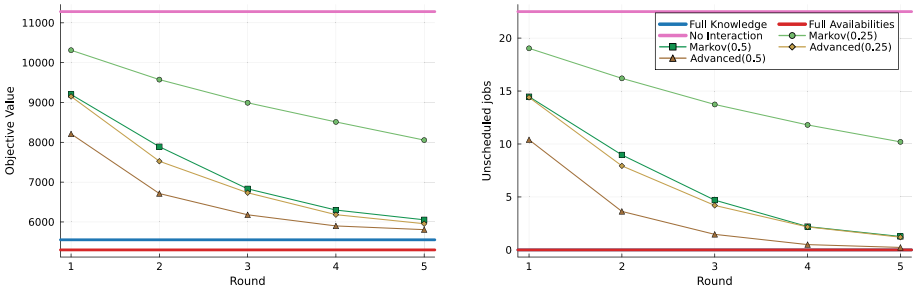



**Fig. 7.** Direct comparison between Markov and Advanced in terms of the objective value and the number of unscheduled jobs for $(m, n, |U|) = (5, 120, 30)$.

performs still quite well in comparison, considering that it does assume almost no knowledge about the users. In particular when looking at the overbooked case in the table, i.e., instances with $n = 48m$, Markov is closer to Advanced and in some cases almost matches the gaps of Advanced.

We furthermore give the median runtime per interaction round for each instance size, aggregated over the different methods and all five iterations. Note that these times are clearly dominated by, and almost match the time for solving the MILP to determine queries. Naturally these times increase with the number of machines with a median time between four and 17 s for one machine and a significant number of

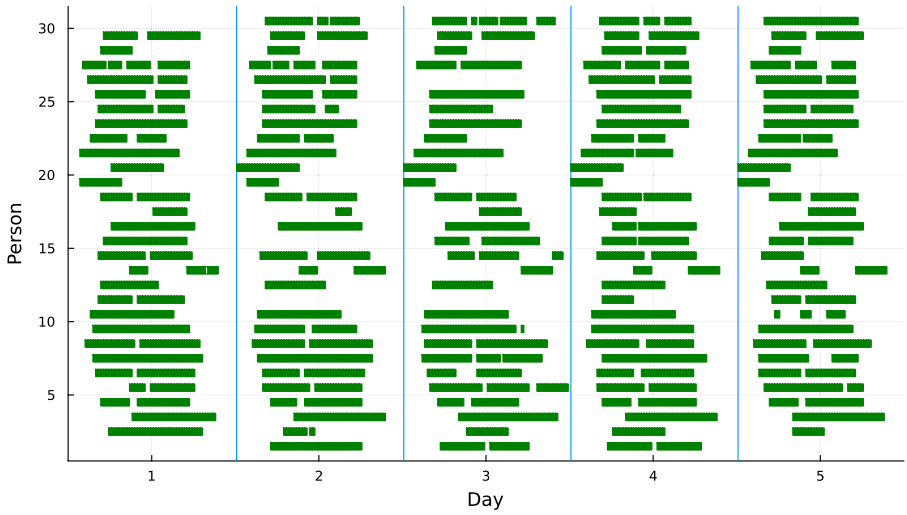



**Fig. 8.** The availabilities derived from the first 30 persons in the filtered Dutch Time-Use-Survey. Days are split by blue vertical lines.

instances hitting the time limit of 15 min for five machines. Note that the instances consistently require more time to solve for more users when $n = 24m$ and less time to solve for more users when $n = 48m$.

### 7.3. Real-world availabilities

To evaluate the generalization abilities of our models to real-world data in the following, we further generated an instance set with $m = 5$ machines, $n = 120$ jobs and $|U| = 30$ users the same way as above, but where the availabilities are derived from the Dutch time-use-survey from 2005 (Sociaal en Cultureel Planbureau, 2005). The reason for using this survey is that it records a full week of the respondents, as opposed to single days for most other time-use-surveys, and that it is publicly available. For each respondent and each 15-minutes timeslot in a week from Sunday to Saturday the survey records the persons activity in categories such as sleeping, work at home, work outside of home, traveling (with a bunch of subcategories), personal care, and so forth. We assume that the user is using the machine at work for some work-related tasks. Thus we assume users to be available when they are present at the workplace, i.e. for the categories "work away from home", "overtime" and "coffee/tea breaks". To make the data fit to our assumptions, we do not consider persons that are unemployed or work less than two hours on at least three days, and persons that work on the weekend or on any day before 6am or after 10pm. This results in a dataset of 490 different sets of availabilities, 30 of them exemplarily visualized in Fig. 8, and they are chosen randomly to generate instances.

Fig. 9 shows the development of the objective value when applying our approaches to these instances. Acceptance probabilities of 0.75 and 0.85 seem to perform best and for those values the objective value quickly converges to the Full Knowledge case, reducing the gap from 127% to 32.3% for Markov(0.75) and to 27.6% for Advanced(0.75) within five rounds. Considering the very different structure of the availability data, the loss of efficiency compared to our completely randomly generated instances apparently is quite small.

### 7.4. Acceptance rate

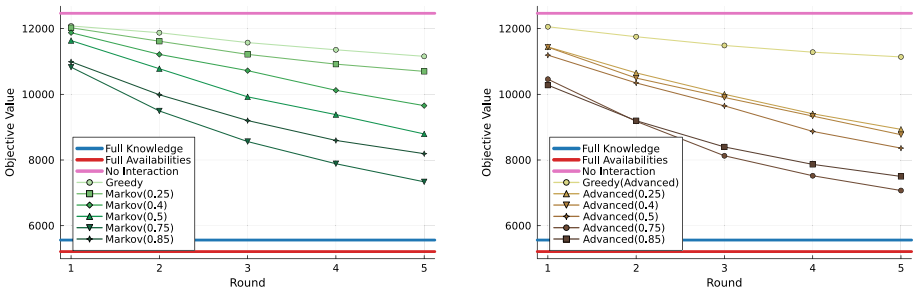
The percentage of queries that have been accepted is of high practical relevance. If users have to reject a majority of queries, they will get frustrated with the system. Fig. 10 shows the mean acceptance rate for Markov and Advanced with different values for $p^{\text{lim}}$ plotted over the rounds for one instance size. As expected, the acceptance rate is higher for higher values of $p^{\text{lim}}$. For Advanced it is a bit higher than $p^{\text{lim}}$, since Advanced incorporates a probability computation that is based on the generation process of the instances, so the MILP only selects from queries that have a probability higher than $p^{\text{lim}}$ to be accepted. In contrast, Markov($p^{\text{lim}}$) has a lower acceptance rate than $p^{\text{lim}}$, as the probabilities are an approximation. Furthermore, there are no significant changes over the individual rounds.

### 7.5. Convergence

Although significantly higher numbers of rounds are in most practical applications not very reasonable, we now study the convergence behaviors of the different approaches over 50 rounds to evaluate the approaches potential capabilities. It will give an insight into how long each approach needs to gather sufficient relevant information for (almost) closing the remaining optimality gap. Instances with $n = 48m$ need longer to converge, thus we show the evolution of the objective values for $(m, n, |U|) = (5, 240, 60)$ in Fig. 11. As Advanced performs better for lower thresholds $p^{\text{lim}}$, we additionally show the evolution for $p^{\text{lim}} = 0.15$ and $p^{\text{lim}} = 0.05$. Surprisingly, Markov performs slightly better when looking at the round by which a gap of less than 5% is reached: With $p^{\text{lim}} = 0.4$, it only requires 14 rounds, while Advanced requires 16 rounds with its best threshold of $p^{\text{lim}} = 0.15$. Both, Markov and Advanced, stagnate with a high $p^{\text{lim}}$ of 0.75 and 0.85. The reason seems to be that after the first rounds they do not gather much new information but mostly just confirm their presumptions when using such a high value for $p^{\text{lim}}$.

### 7.6. Influence of the number of users

We now investigate the impact of varying the number of jobs per user, and thus the number of users. Fig. 12 compares obtained objective values after five rounds for instances with different numbers of users. A higher number of jobs per user (or lower number of users) make the selection of user queries more challenging—for two jobs per user even Greedy performs reasonably well with a mean gap of 22.2% to the Full Knowledge case after five rounds, opposed to a gap of 80.3% for six jobs per user. In contrast Markov(0.75) has similar gaps of 10.2% and 12.6% for two and six jobs per user. The same behavior is also apparent for Advanced with gaps of 4.63% and 7.01% for two and six jobs per user and $p^{\text{lim}} = 0.5$. We explain these differences with the higher total amount of interaction that comes with a higher number of users.




(a) Markov (b) Advanced

**Fig. 9.** Development of the objective value when applying (a) Markov and (b) Advanced with different acceptance thresholds for the instances based on the Dutch Time-Use-Survey, $(m, n, |U|) = (5, 120, 30)$.

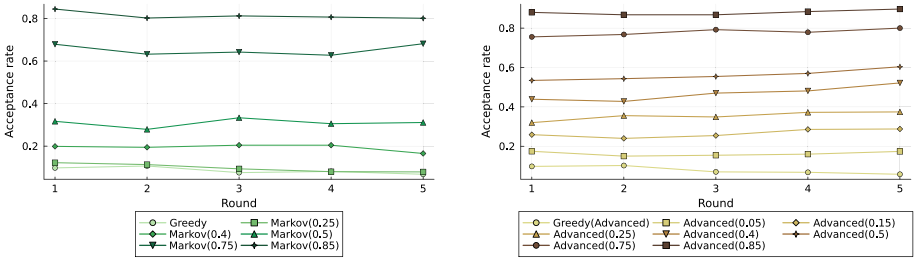



**Fig. 10.** Mean acceptance rate of the users over the rounds for instances with $(m, n, |U|) = (5, 120, 30)$ and the two models to calculate probabilities.

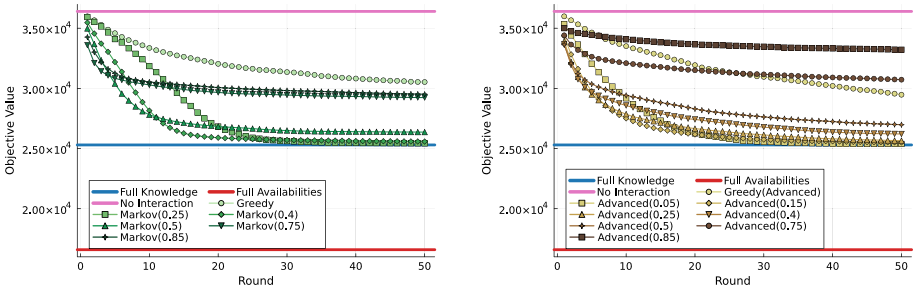



**Fig. 11.** Convergence behavior of Markov and Advanced for $(m, n, |U|) = (5, 240, 60)$.

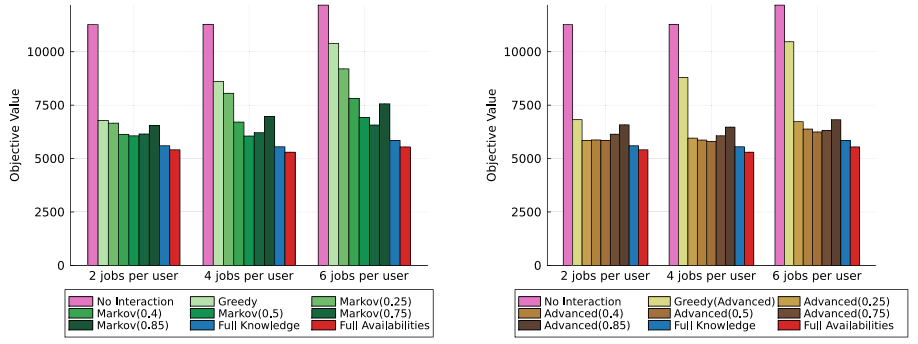



**Fig. 12.** Impact of the number of users to the objective value after five rounds of user interaction for Markov and Advanced and instances with $(m, n) = (5, 120)$ and 2, 4 and 6 jobs per user, respectively.

### 7.7. Comparison of different interaction configurations

We further investigate the impact of different levels of user interaction by varying $b$ and letting users suggest more than one starting time for each job; denote with $n^{\text{prop}}$ the number of initially suggested starting times per job. For this purpose we iteratively add two additionally proposed starting times for each job to the instances with $m = 5$, $n \in \{120, 240\}$ and $|U| = n/4$, leading to instances with $n^{\text{prop}} \in \{1, 2, 3\}$. We perform simulations on these instances with an interaction budget $b$ of $0.5|U|$, $|U|$ and $2|U|$. The results are summarized in Table 2.

Fig. 13 compares the results for different interaction budgets $b \in \{0.5|U|, |U|, 2|U|\}$ per round. Naturally a higher budget results in faster convergence. However, differences are relatively small, and $b = 0.5|U|$ does not lead to significantly worse results. When comparing gaps after a fixed amount of user interactions, a lower number of interactions per round performs better. For example, the mean gap for Markov(0.5)

**Table 2**
Mean %-gaps of the objective values after five interaction rounds for Markov($p^{\text{lim}}$) and Advanced($p^{\text{lim}}$) with different limits $p^{\text{lim}}$ and for Greedy and Greedy(Advanced) for varying values of $b$ and $n^{\text{prop}}$. The median runtime per interaction round is given for each instance size, aggregated over methods and iterations rounds.

| $m$ | $n$ | $\lvert U\rvert$ | $b$ | $n^{\text{prop}}$ | Markov | | | | | | Advanced | | | | | | Time [s] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Greedy | 0.25 | 0.4 | 0.5 | 0.75 | 0.85 | Greedy | 0.25 | 0.4 | 0.5 | 0.75 | 0.85 | |
| 5 | 120 | 30 | 15 | 1 | 83.2 | 65.4 | 34.3 | 21.4 | **20.5** | 39.5 | 81.7 | 16.0 | 10.4 | **8.2** | 12.4 | 21.9 | 903.6 |
| 5 | 120 | 30 | 15 | 2 | 8.5 | 7.4 | 5.3 | 4.3 | **3.4** | 4.2 | 8.4 | 3.2 | 2.9 | **2.7** | 4.9 | 6.1 | 904.8 |
| 5 | 120 | 30 | 15 | 3 | 4.1 | 3.7 | 3.1 | 2.4 | **1.6** | 1.9 | 4.1 | 1.4 | **1.3** | 1.5 | 3.1 | 3.6 | 137.1 |
| 5 | 120 | 30 | 30 | 1 | 56.7 | 47.1 | 21.9 | **8.9** | 12.3 | 25.6 | 58.8 | 7.5 | 5.5 | **5.1** | 9.4 | 16.0 | 912.9 |
| 5 | 120 | 30 | 30 | 2 | 6.6 | 6.2 | 3.9 | **2.5** | 2.9 | 3.4 | 6.3 | **1.8** | 2.0 | 2.3 | 4.6 | 6.0 | 271.6 |
| 5 | 120 | 30 | 30 | 3 | 3.5 | 3.1 | 2.1 | **1.4** | 1.4 | 1.7 | 3.6 | **1.0** | 1.1 | 1.3 | 3.0 | 3.5 | 135.6 |
| 5 | 120 | 30 | 60 | 1 | 16.7 | 15.7 | 6.6 | **4.8** | 6.8 | 13.7 | 16.5 | 3.5 | **2.8** | 3.2 | 6.8 | 12.5 | 174.6 |
| 5 | 120 | 30 | 60 | 2 | 5.0 | 3.9 | 2.4 | **1.8** | 2.4 | 2.9 | 4.8 | **1.2** | 1.5 | 2.0 | 4.5 | 5.8 | 124.4 |
| 5 | 120 | 30 | 60 | 3 | 3.0 | 2.7 | 1.7 | **1.1** | 1.4 | 1.6 | 3.1 | **0.8** | 0.9 | 1.2 | 2.9 | 3.5 | 117.6 |
| 5 | 240 | 60 | 30 | 1 | 43.2 | 40.8 | 34.6 | 29.0 | **24.1** | 26.8 | 43.1 | 22.3 | **21.9** | 23.0 | 30.5 | 37.2 | 162.6 |
| 5 | 240 | 60 | 30 | 2 | 29.2 | 27.4 | 23.9 | 20.2 | **14.5** | 16.5 | 29.1 | 13.2 | **13.2** | 14.2 | 22.6 | 26.8 | 106.6 |
| 5 | 240 | 60 | 30 | 3 | 22.6 | 21.4 | 19.1 | 15.6 | **10.8** | 12.7 | 22.7 | **9.3** | 9.7 | 10.9 | 18.8 | 21.3 | 127.2 |
| 5 | 240 | 60 | 60 | 1 | 37.2 | 35.3 | 27.7 | **20.5** | 22.3 | 23.9 | 37.4 | **16.9** | 17.6 | 20.3 | 29.4 | 36.7 | 199.2 |
| 5 | 240 | 60 | 60 | 2 | 26.5 | 25.7 | 20.2 | 13.5 | **13.5** | 15.2 | 26.7 | **10.8** | 11.3 | 13.0 | 22.1 | 26.5 | 158.2 |
| 5 | 240 | 60 | 60 | 3 | 21.3 | 19.9 | 15.5 | 10.8 | **9.8** | 11.8 | 21.3 | **7.4** | 8.1 | 9.8 | 18.4 | 21.1 | 146.6 |
| 5 | 240 | 60 | 120 | 1 | 29.3 | 27.1 | 18.1 | **15.3** | 21.1 | 23.3 | 29.4 | **14.0** | 15.6 | 18.6 | 28.9 | 36.3 | 141.1 |
| 5 | 240 | 60 | 120 | 2 | 22.8 | 20.4 | 14.6 | **10.9** | 12.7 | 14.9 | 22.4 | **9.0** | 10.2 | 11.8 | 21.8 | 26.3 | 136.3 |
| 5 | 240 | 60 | 120 | 3 | 18.7 | 16.9 | 12.2 | **8.8** | 9.5 | 11.5 | 18.7 | **6.5** | 7.5 | 9.2 | 18.0 | 21.1 | 126.9 |

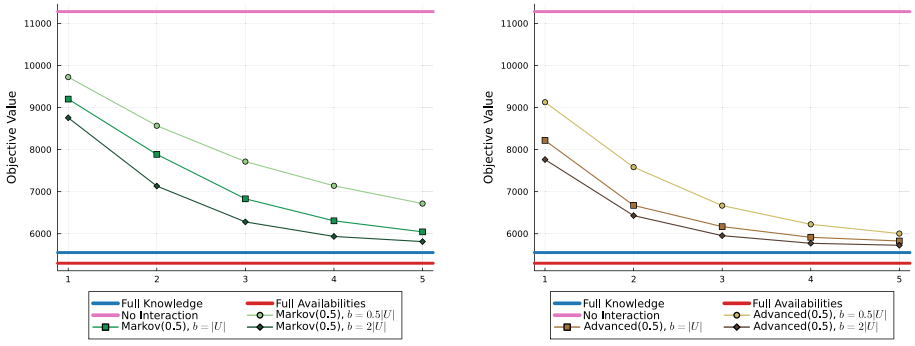
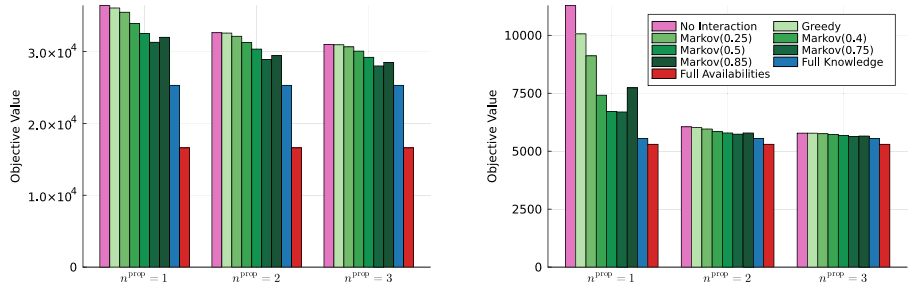



**Fig. 13.** Comparison between different interaction budgets $b \in \{0.5|U|, |U|, 2|U|\}$ for instances with $(m, n, |U|) = (5, 120, 30)$.




(a) $(m, n, |U|) = (5, 240, 60)$. (b) $(m, n, |U|) = (5, 120, 30)$.

**Fig. 14.** Comparison between different values of $n^{\text{prop}} \in \{1, 2, 3\}$ with interaction budget $b = 0.5|U|$ for two different instance sizes.

with $b = 0.5|U|$ after four rounds is 29.2%, with $b = |U|$ after two rounds 42.5%, and with $b = 2|U|$ after one round 58.6%, while in each case $2|U|$ replies were requested from the users. This indicates that our approach is indeed capable to learn from and react to the user's replies—successively produced queries are more cleverly chosen. When just considering the number of user interactions, it would thus be more user-friendly to make more interaction rounds with in total less queries, asking only a small subset of users in each round, and the final result can still be of similar quality. However, note that depending on the specific application and the form by which the user interactions are actually implemented, also the number of performed interaction rounds may substantially impact the perceived user-friendliness.

A comparison for different values of $n^{\text{prop}}$ for $b = 0.5|U|$ is shown in Fig. 14. It becomes apparent that, for $n = 48m$, $n^{\text{prop}} = 2$ without any interaction leads to a mean gap of 29.4%, which is worse than for $n^{\text{prop}} = 1$ with five rounds of interaction, which leads to a mean gap of 24.1% with Markov(0.75). Specifying an additional starting time for each of their four jobs is clearly more bothersome for the users than answering two to three queries, still the interactive approach gives a better result. A value of $n^{\text{prop}} = 3$ leads to a better gap of 22.8%, at the cost of bothering the users even more. The situation is different for $n = 24m$. No interaction with $n^{\text{prop}} = 2$ then leads to a gap of 9.08%, which is clearly better than the gap with $n^{\text{prop}} = 1$ after five rounds of interaction, which is 20.5%. One more starting time for each job ($n^{\text{prop}} = 3$) or, alternatively, five rounds of interaction with

MARKOV(0.75) further more than halves the gap to 4.14% and 3.35%, respectively.

## 8. Conclusions

We considered the problem of scheduling jobs involving humans, whose availabilities can only be partially revealed with few time interval queries, made in a small number of interaction rounds. The proposed solution approach calculates probabilities for users to accept suggested time intervals based on a two-state Markov process or, alternatively, a more advanced Markov process that is based on more specific assumptions of user behavior. An ILP is used as optimization core and to select time intervals for the next round of queries, aiming for sufficiently high probabilities of acceptance and a maximum cost reduction. Experiments on artificial test instances show that an initial solution quickly improves over the interaction rounds and may soon get close to a solution of the full-knowledge case despite the very restricted interaction. With the variant of our approach that utilizes the advanced Markov process, we demonstrated how the general approach can be specialized when a more specific stochastic model for the user availabilities is available. While this advanced variant performs significantly better on our benchmark instances, also the simpler and more generally applicable approach based on the two-state Markov process compares surprisingly well. Still, the results suggest that more knowledge about the users is beneficial, and in future work it would be interesting to include and utilize historic user availability data, which reflects the users' general preferences. Moreover, alternative ways to consider the estimated acceptance probabilities of user queries in the optimization core should be investigated.

## CRediT authorship contribution statement

**Johannes Varga:** Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Günther R. Raidl:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Funding acquisition, Conceptualization. **Elina Rönnberg:** Writing – review & editing, Writing – original draft, Conceptualization. **Tobias Rodemann:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

## Data availability

A link to our test instances is provided in the paper.

## Acknowledgments

J. Varga acknowledges the financial support from Honda Research Institute Europe. The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

## References

Aghaei-Pour, P., Rodemann, T., Hakanen, J., Miettinen, K., 2022. Surrogate assisted interactive multiobjective optimization in energy system design of buildings. Opt. Eng. 23 (1), 303–327.

Anghinolfi, D., Paolucci, M., Ronco, R., 2021. A bi-objective heuristic approach for green identical parallel machine scheduling. European J. Oper. Res. 289 (2), 416–434.

Blum, A., 1997. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Mach. Learn. 26 (1), 5–23.

Chen, B., Zhang, X., 2019. Scheduling with time-of-use costs. European J. Oper. Res. 274 (3), 900–908.

Cheng, J., Chu, F., Zhou, M., 2018. An improved model for parallel machine scheduling under time-of-use electricity price. IEEE Trans. Autom. Sci. Eng. 15 (2), 896–899.

Ding, J.-Y., Song, S., Zhang, R., Chiong, R., Wu, C., 2016. Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. IEEE Trans. Autom. Sci. Eng. 13 (2), 1138–1154.

Graham, R., Lawler, E., Lenstra, J., Kan, A., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Hammer, P., Johnson, E., Korte, B. (Eds.), Discrete Optimization II. In: Annals of Discrete Mathematics, vol. 5, Elsevier, pp. 287–326.

Hammack, R., Imrich, W., Klavžar, S., 2011. Handbook of product graphs, second edition. CRC Press, pp. 1–536.

Jatschka, T., Raidl, G.R., Rodemann, T., 2021. A general cooperative optimization approach for distributing service points in mobility applications. Algorithms 14 (8).

Kayhan, B.M., Yildiz, G., 2021. Reinforcement learning applications to machine scheduling problems: a comprehensive literature review. J. Intell. Manuf. 34, 909–929.

Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J., Zabowski, D., et al., 1994. Experience with a learning personal assistant. Commun. ACM 37 (7), 80–91.

Monaci, M., Agasucci, V., Grani, G., 2024. An actor-critic algorithm with policy gradients to solve the job shop scheduling problem using deep double recurrent agents. European J. Oper. Res. 312 (3), 910–926.

Saberi-Aliabad, H., Reisi-Nafchi, M., Moslehi, G., 2020. Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. J. Clean. Prod. 249, 119393.

Saha, S., Minku, L.L., Yao, X., Sendhoff, B., Menzel, S., 2021. Exploiting linear interpolation of variational autoencoders for satisfying preferences in evolutionary design optimization. In: 2021 IEEE Congress on Evolutionary Computation. IEEE Press, pp. 1767–1776.

Sociaal en Cultureel Planbureau, 2005. Tijdsbestedingsonderzoek 2005 - TBO 2005. http://dx.doi.org/10.17026/dans-znn-5xvz, DANS Data Station Social Sciences and Humanities.

Uzunoglu, A., Gahm, C., Wahl, S., Tuma, A., 2023. Learning-augmented heuristics for scheduling parallel serial-batch processing machines. Comput. Oper. Res. 151, 106122.

Varga, J., Karlsson, E., Raidl, G.R., Rönnberg, E., Lindsten, F., Rodemann, T., 2023a. Speeding up logic-based benders decomposition by strengthening cuts with graph neural networks. In: Nicosia, G., Ojha, V., La Malfa, E., La Malfa, G., Pardalos, P., Umeton, R. (Eds.), Machine Learning, Optimization, and Data Science. LOD 2023. Lecture Notes in Computer Science. vol. 14505, Springer, Cham, pp. 24–38.

Varga, J., Raidl, G.R., Rönnberg, E., Rodemann, T., 2023b. Interactive job scheduling with partially known personnel availabilities. In: Dorronsoro, B., Chicano, F., Danoy, G., Talbi, E.-G. (Eds.), OLA 2023: Optimization and Learning. In: Communications in Computer and Information Science, vol. 1824, Springer, pp. 236–247.

Wang, S., Wang, X., Yu, J., Ma, S., Liu, M., 2018. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. J. Clean. Prod. 193, 424–440.

Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., Hu, K., 2012. Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. Comput. Oper. Res. 39 (7), 1315–1324.