Interactive Job Scheduling with Partially Known Personnel Availabilities*

Johannes Varga[1**][0000−0003−1413−7115], Günther R. Raidl[1][0000−0002−3293−177X], Elina Rönnberg[2][0000−0002−2081−2888], and Tobias Rodemann[3][0000−0001−6256−0060]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{jvarga,raidl}@ac.tuwien.ac.at
[2] Department of Mathematics, Linköping University, Linköping, Sweden
elina.ronnberg@liu.se
[3] Honda Research Institute Europe, Offenbach, Germany
tobias.rodemann@honda-ri.de

Abstract. When solving a job scheduling problem that involves humans, the times in which they are available must be taken into account. For practical acceptance of a scheduling tool, it is further crucial that the interaction with the humans is kept simple and to a minimum. Requiring users to fully specify their availability times is typically not reasonable. We consider a scenario in which initially users only suggest single starting times for their jobs and an optimized schedule shall then be found within a small number of interaction rounds. In each round users may only be suggested a small set of alternative time intervals, which are accepted or rejected. To make the best out of these limited interaction possibilities, we propose an approach that utilizes integer linear programming and a theoretically derived probability calculation for the users' availabilities based on a Markov model. Educated suggestions of alternative time intervals for performing jobs are determined from these acceptance probabilities as well as the optimization's current state. The approach is experimentally evaluated and compared to diverse baselines. Results show that an initial schedule can be quickly improved over few interaction rounds, and the final schedule may come close to the solution of the full-knowledge case despite the limited interaction.

Keywords: Job scheduling · human machine interaction · preference learning · integer linear programming

1 Introduction

We consider a class of job scheduling problems in which the personnel of a company is involved as a bottleneck resource. The central aim is to schedule jobs of employees in an interactive way that works from the humans' perspectives

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

as simple, stress-free, and with low cognitive effort—while at the same time a cost function is minimized. In the simplest, and from the users' perspective most convenient case, each user just suggests a starting time for each of her or his jobs. As the jobs also require further shared resources, this directly obtained schedule will rarely be feasible nor are cost-aspects considered. Ideally, we would have full knowledge of all the users' times at which they would be available for performing their jobs, in which case we could solve an optimization problem in one shot. In many practical scenarios, however, it is impossible or far too inconvenient to request such complete information. We therefore start with the users' initial suggestions and perform a small number of simple *interaction rounds* to get more freedom for finding better schedules. In each such round the solution approach is allowed to suggest each user a small number of additional time intervals for scheduling her or his jobs. The users are then supposed to indicate their acceptance or rejection of these intervals. Hereby, we intentionally avoid that users are requested to specify larger amounts of additional availability intervals on their own. With the increased knowledge on the users' availabilities, the optimization can aim at improving the solution in each round.

The main challenge we address in this work is to, in each round, come up with *meaningful queries* for further time intervals to perform jobs in. Queried time intervals are most meaningful when (a) they would allow the optimization to obtain a better schedule and (b) the users are likely to accept them. For example, very large time intervals may aid the optimization the most, but as they are rather unlikely to be accepted by the users, they are usually not that meaningful. To consider (b), 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 precise user-specific stochastic models available, for example derived from historic availability data. Here we assume that such information is not available and instead build upon just a simple stochastic model represented by a two-stage Markov process. In essence, we only assume to know average probabilities of users to be available/unavailable in a timestep under the condition that the user is known to be available/unavailable in the directly preceding timestep.

The overall scenario can also be seen as *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, (b) to narrow it down to a specific *Interactive Job Scheduling Problem* (IJSP) to make concrete computational investigations on, (c) an Integer Linear Programming (ILP) model as optimization core for solving the IJSP, (d) an exact and computationally efficient calculation of the probabilities for users to accept potentially queried time intervals based on the two-stage Markov process and the already known availability information from the users, and (e) to propose a heuristic solution approach for the IJSP that utilizes this probability calculation. 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 simplistic assumptions of the two-stage Markov model, schedules may be obtained that come close to those of solving the full-knowledge case.

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 bought on the spot market, (expected) electricity costs may change significantly over time. For avoiding to have 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 core of this problem, if neglecting the users, can be described in the common three-field notation for scheduling problems as Pm||TEC, where Pm refers to the m machines and that job durations do not depend on the machines, and where the objective is to minimize the Total Energy Costs (TEC). The similar problem Pm||$C_{\max}$,TEC, which additionally takes the makespan into account for the objective, has been considered in the literature. Solving approaches for it include a Mixed Integer Linear Program (MILP) [10,2], a problem specific heuristic and a genetic algorithm [10], as well as a greedy heuristic and local search [2]. Also similar is the scheduling problem Rm||TEC where jobs have in general different processing times on different machines. For it, Ding et al. [5] proposed a MILP and a Dantzig-Wolfe decomposition. The MILP was further improved by Cheng et al. [4] and by Saberi-Aliabad et al. [8].

In interactive optimization approaches, most works only consider a single user who guides the optimization process. For instance, Saha et al. [9] develop approaches based on evolutionary algorithms that cooperate with human designers to find aesthetic, aerodynamic, and structurally efficient designs for automotives, and Aghaei-Pour et al. [1] consider a multiobjective optimization problem where the human interactively specifies preferences on the solution, which are also considered within evolutionary algorithms. Interactive optimization with multiple users is less common. For instance, Jatschka et al. [6] consider a MILP-based cooperative optimization approach that interacts with many users to learn an objective function for distributing service points in mobility applications.

We perform active learning on the availability times of the users. This has also 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 [7], the weighted-majority algorithm or the Winnow algorithm [3] for the learning task.

The next section formalizes our IJSP and introduces the ILP used as optimization core. Section 3 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 based on a two-stage Markov model is subsequently detailed in Section 4. Section 5 shows experimental results, and Section 6 concludes this work.

2 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 $t^{\max}$ uniform timesteps, and let $T = \{t \mid t = (t^{\text{day}}, t^{\text{time}}),\ t^{\text{day}} = 1, \dots, t^{\text{max-day}},\ t^{\text{time}} = 1, \dots, t^{\max}\}$ 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 $t_1, t_2 \in T \mid t_1^{\text{day}} = t_2^{\text{day}},\ t_1^{\text{time}} \leq t_2^{\text{time}}$, and adding a scalar Δ 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 $d_j \in \{1, \dots, t^{\max}\}$ 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 possible starting times of job $j \in J$ are restricted to the set $T_j^{\text{job}} = \bigcup_{t^{\text{day}}=1}^{t^{\text{max-day}}} \{(t^{\text{day}}, 1), \dots, (t^{\text{day}}, t^{\max} - d_j + 1)\}$, because of the job duration. 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 availability of 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 time-dependent cost $c_{it} \geq 0$, 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 its user and a machine for the complete duration of the job. If a job $j \in J$ cannot be feasibly scheduled, this induces cost $q_j \geq 0$, 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., $q_j \geq d_j \max_{i \in M,\, t \in T} c_{it},\ j \in J$.

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. This can be expressed by 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.

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

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

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

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

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

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^{avail} and T_j^{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^{rej}. In such update, I_u^{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 denoted by $B \in \mathbb{N}_{>0}$, and let the allowed number of queries in each round be $b \in \mathbb{N}_{>0}$. In each round, a user may be queried multiple times. 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.

3 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 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

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

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

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 $[t, t + d_j]$ for user u. We refer to this approach to determine user queries by Greedy.

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^{query} whose associated queries have probabilities below a given threshold $0 \leq p^{\text{lim}} \leq 1$, 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^{query}. As model for the acceptance probabilities, the next section proposes one based on a two-state Markov process, and consequently, we refer to this advanced model-based solution approach by Markov(p^{lim}).

4 Probability Calculation for Two-State Markov Process

Consider a single user $u \in U$ and a single day $t^{\text{day}} \in \{1, \ldots, t^{\text{max-day}}\}$. For better readability we refer to the timesteps of this day in the following by $T_{t^{\text{day}}} = \{1, \ldots, t^{\text{max}}\}$. 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 ρ_{00}, ρ_{01}, ρ_{10}, and ρ_{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 Figure 1a. 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^{avail} and the set of so far rejected time intervals I^{rej}, we now want to determine the probability that the user is available in some given time interval $[\tau, \tau']$, $1 \leq \tau \leq \tau' \leq t^{\text{max}}$. For this

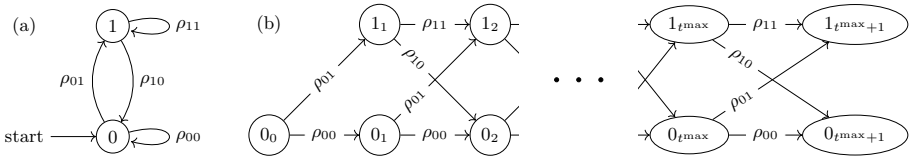


Fig. 1: (a) Two-state Markov process and (b) corresponding unrolled state graph.

purpose we unroll the Markov process into a state graph over all timesteps from 0 to $t^{\max}+1$ as follows and illustrated in Figure 1b.

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 $0_{t^{\max}+1}$ and for now $1_{t^{\max}+1}$ 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 ρ_{00}, ρ_{01}, ρ_{10}, and ρ_{11}.

Ignoring known user availabilities T^{avail} and rejected time intervals I^{rej} for now, this state graph has been constructed in such a way that each path from node 0_0 to either node $0_{t^{\max}+1}$ or $1_{t^{\max}+1}$, which we call terminal nodes, corresponds to exactly one outcome of the Markov process over $t^{\max}+1$ 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^{avail} of the user by removing all nodes 0_t for T^{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^{avail}. Moreover, we also remove node $1_{t^{\max}+1}$ with its ingoing arcs in order to model that the user is unavailable after the last actual timestep $t^{\max}$.

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

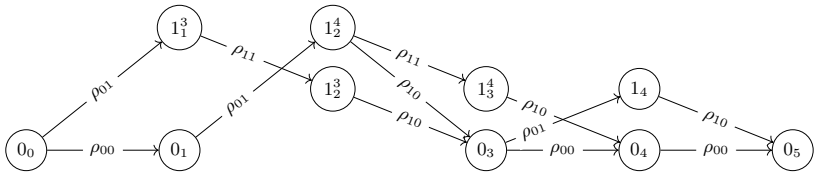


Fig. 2: The state graph for $t^{\max} = 4$, $T^{\text{avail}} = \{2\}$, and $I_u^{\text{rej}} = \{(1,3),(2,4)\}$.

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 replaced by arcs $(0_t, 1_{t+1}^{t_2})$ and $(1_t, 1_{t+1}^{t_2})$, 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^{rej} has been guaranteed not to contain a proper subinterval of $[t_1, t_2]$. Each new node $1_t^{t_2}$ 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 $1_t^{t_2}$ to node $1_{t+1}^{t_2}$ as long as $t + 1 < t_2$ for the case of staying in state 1. Due to the absence of an arc from node $1_{t_2-1}^{t_2}$ 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 Figure 2.

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 $p_{0_0,v}^{\text{path}}$ of all paths from 0_0 to v, denoted by $\text{Paths}(0_0, v)$, can be computed recursively as

$$\begin{aligned} p_{0_0,v}^{\text{path}} &= \sum_{P \in \text{Paths}(0_0,v)} \prod_{(u,u') \in P} \rho(u,u') \\ &= \sum_{u \in N^-(v)} \sum_{P \in \text{Paths}(0_0,u)} \left(\prod_{(u,u') \in P} \rho(u,u') \right) \cdot \rho(u,v) = \sum_{u \in N^-(v)} p_{0_0,u}^{\text{path}} \rho(u,v), \end{aligned} \quad (8)$$

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 $p_{v,0_{t^{\max}+1}}^{\text{path}}$ of all paths from a node v to node $0_{t^{\max}+1}$ can be correspondingly computed recursively by

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

We are now interested in all those paths from 0_0 to $0_{t^{\max}+1}$ that stay for the timesteps τ to τ' in state 1 nodes, indicating the availability of the user. Each of these paths is composed of a path from 0_0 to $1_\tau^{t_2}$, a path P from $1_\tau^{t_2}$ to $1_{\tau'}^{t_2}$ 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_τ 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 τ to τ' 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^{avail} and I^{rej} and ending in $0_{t^{\max}+1}$, is of main interest, which is

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

where the sum is taken over all middle segments 1-Paths(τ, τ'), and P_τ 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 τ to τ'.

5 Experimental Evaluation

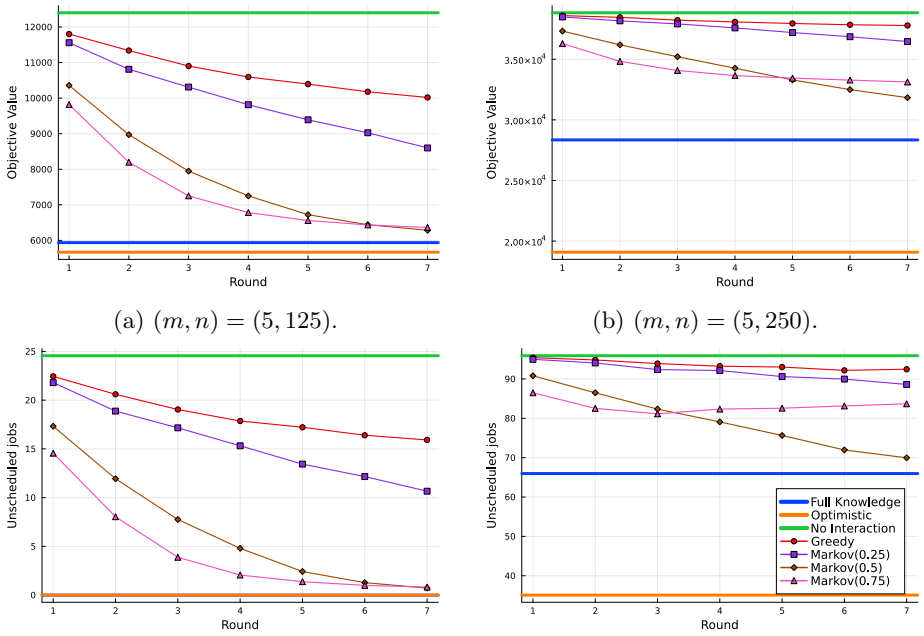
We implemented the approaches in Julia 1.8.3, using the solver Gurobi 10.0 (https://www.gurobi.com) and the package JuMP as interface to Gurobi. As real world instances were not available to us we created artificial benchmark instances and used them to compare the approaches with each other. Each test run was performed on a single core of an AMD EPYC 7402 and Gurobi was given a timelimit of 15 minutes for each ILP, which always led to final gaps below 5%.

5.1 Instance Generation

We consider a time horizon of $t^{\text{max-day}} = 5$ days, each starting at 6am and ending at 10pm, with a time granularity of 15 minutes 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 μ^{start} and standard deviation σ^{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 μ^{dur} and standard deviation σ^{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.

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

For each job $j \in J_u$ of a user $u \in U$ the duration d_j is chosen uniformly at random from 30min to 4h. Moreover, a starting time t_j is selected at random so



(a) $(m, n) = (5, 125)$. (b) $(m, n) = (5, 250)$.

(c) $(m, n) = (5, 125)$. (d) $(m, n) = (5, 250)$.

Fig. 3: Development of the objective value (a) and (b) respectively number of unscheduled jobs (c) and (d) for two different instance sizes.

that the job can in principle be scheduled within $T_u^{\text{avail}*}$. The initially provided set of availabilities for user $u \in U$ is then $T_u^{\text{avail}} = \bigcup_{j \in J_u} T_j[t_j]$.

We generate 50 instances for $m \in \{1, 2, 3, 4, 5\}$ machines and either 25 or 50 jobs per machine $n \in \{25m, 50m\}$. When considering the generated user availabilities, each machine can execute roughly 30 jobs on average, thus for $n = 25m$ usually it is possible to schedule all jobs, while for $n = 50m$ this is not the case. Each user has five jobs, thus there are either $5m$ or $10m$ users. We allow $|U|$ user queries in each round, for a total of seven rounds.

The costs are based on the real-world spot market prices c_t^{kWh} for electricity in Germany from week 26 in 2022 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 [50kW,150kW]. 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.

5.2 Comparison of the Approaches

We performed simulations for Greedy and Markov(p^{lim}) with acceptance probability thresholds $p^{\text{lim}} \in \{0.25, 0.5, 0.75\}$ on all benchmark instance. After each round we determine the best schedule that is feasible for the information

Table 1: Mean %-gaps of the objective values after five and seven interaction rounds for GREEDY and MARKOV(p^{lim}) with different limits p^{lim}.

m	n	Round 5				Round 7			
		GREEDY	MARKOV(p^{lim})			GREEDY	MARKOV(p^{lim})		
			0.25	0.5	0.75		0.25	0.5	0.75
1	25	77.7	54.5	**39.6**	40.0	70.7	36.9	**18.0**	24.6
2	50	95.2	74.5	27.0	**23.8**	87.0	52.6	**11.9**	13.3
3	75	77.5	62.3	18.2	**15.0**	69.1	48.8	**8.4**	9.3
4	100	79.5	64.3	15.7	**12.1**	71.8	47.9	**6.7**	7.9
5	125	77.8	60.4	13.4	**10.7**	71.5	46.7	**6.0**	7.3
1	50	40.2	33.9	**19.4**	22.6	37.1	29.5	**12.8**	21.7
2	100	36.8	31.9	**18.6**	19.2	35.6	29.1	**13.0**	18.2
3	150	34.4	31.6	**17.8**	18.0	33.4	28.7	**12.1**	17.0
4	200	35.0	32.2	**18.3**	18.8	34.0	29.2	**12.6**	17.6
5	250	34.4	31.8	**17.7**	18.3	33.8	29.1	**12.4**	17.2

collected up to this round. Figure 3 shows the development of the mean objective value and mean number of unscheduled jobs, respectively, over the rounds. Values are aggregated over the 50 instances with $m = 5$ machines and $n = 125$ respectively $n = 250$ jobs. Furthermore, we determine the best feasible schedule with the information that is available before the first round ("No Interaction"), the best schedule when ignoring user availabilities ("Optimistic") 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 and MARKOV(p^{lim}) in respect to "Full Knowledge" after five and seven rounds in percent.

We observe that MARKOV(0.5) and MARKOV(0.75) quickly converge towards the best possible schedule. For $n = 125$, the original objective values without interaction could almost be halved after already five rounds, while for $n = 250$, 18% and 15% of the original costs could be saved after seven rounds. Moreover, for $n = 125$, the final optimality gaps of these two approaches are by a factor of more than nine 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 35%. 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.

6 Conclusions

We considered a job scheduling problem in which humans are involved as resource and where their availabilities can only be partially revealed in a small number of interaction rounds, within which few time interval queries can be

made. The proposed solution approach calculates probabilities for users to accept suggested time intervals based on a two-state Markov process. 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 and the simple assumptions of the two-state Markov process. In future work it would be interesting to replace the proposed probability computation by a machine learning model trained on historic user availability data. Moreover, alternative ways to consider the estimated acceptance probabilities of user queries in the optimization core should be investigated.

References

1. Aghaei-Pour, P., Rodemann, T., Hakanen, J., Miettinen, K.: Surrogate assisted interactive multiobjective optimization in energy system design of buildings. Optimization and Engineering **23**(1), 303–327 (2022)
2. 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)
3. Blum, A.: Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Machine Learning **26**(1), 5–23 (1997)
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. 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)
6. Jatschka, T., Raidl, G.R., Rodemann, T.: A general cooperative optimization approach for distributing service points in mobility applications. Algorithms **14**(8) (2021)
7. 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)
8. 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)
9. Saha, S., Minku, L.L., Yao, X., Sendhoff, B., Menzel, S.: Exploiting linear interpolation of variational autoencoders for satisfying preferences in evolutionary design optimization. In: 2021 IEEE Congress on Evolutionary Computation. pp. 1767–1776 (2021)
10. 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)