# Selecting User Queries in Interactive Job Scheduling*

Johannes Varga[1], Günther R. Raidl[1], and Tobias Rodemann[2]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{jvarga,raidl}@ac.tuwien.ac.at
[2] 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 and extend a scenario from the literature 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 a small amount of information can be requested by suggesting alternative time intervals, which are accepted or rejected. We extend the scenario by another form of interaction that allows to request users to indicate alternative time intervals for their jobs. To make the best out of these limited interaction possibilities, we propose a stochastic programming approach that utilizes a Markov model to consider the users' availabilities. The approach is experimentally evaluated and compared to the approach from the literature. Results show that the stochastic programming approach performs significantly better than the former method from the literature, especially when being able to request alternative time intervals from users.

## 1 Introduction

We consider the Interactive Job Scheduling Problem (IJSP) [6], in which human users, e.g., the personnel of a company, need to perform jobs on some shared machines and the availabilities of these users as well as the machines is critical. In such situations, it is rarely practical to ask users to fully specify their availability times. Instead, we assume 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 make a small number of queries to users to find out more about their availabilities. We investigate two different forms of queries. For the first one, the approach proposes alternative time intervals for the users' jobs, which the users either accept or reject, and for the second, users

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

are asked to provide alternative possible starting times for their jobs restricted to a timeframe. We use a stochastic programming approach to find the queries in each round that have the highest potential to improve the schedule. To formulate expected costs, subject to the selected queries, we use two different stochastic user models. The first one assumes that the users availability in a timestep only depends on their availability in the previous timestep, and the second one assumes that the user is available in up to two intervals each day with normally distributed interval endpoints.

The IJSP has already been considered in the literature by Varga et al. [5, 6]. They only use the first, simpler, type of user query and select queries in each round by calculating an acceptance probability for each query, discarding low-probability queries and selecting those that improve the objective value most. This strategy leads to good convergence towards the best achievable schedule. Our contributions extend these works by considering an additional query type and proposing a stochastic programming approach to get more relevant information from the same amount of interaction.

We are not aware of any other work considering the IJSP or a similar setting. Some approaches to solving a scheduling problem similar to our core problem without the interaction aspect include Mixed Integer Linear Programming [7, 1], a genetic algorithm [7], as well as a greedy heuristic and local search [1]. Moreover, there is a rich literature on human-machine cooperation ranging from cooperative optimization approaches, see e.g. Jatschka et al. [2], to the measuring of the level of cooperation, as done for instance by Wiebel et al. [8].

The next section defines the problem formally and introduces the notation used throughout the paper. Afterwards, Section 3 describes the user models and how they are used to calculate probabilities and to generate samples of user availabilities. Subsequently, Section 4 discusses our stochastic programming approach and Section 5 presents the results of an experimental evaluation. Finally, Section 6 concludes the work and gives an outlook on future work.

## 2 Problem Formulation and Notation

The time planning horizon $T$ of the IJSP consists of multiple days, each with the same number of discrete time steps. We denote with $U$ the set of users. Each user $u \in U$ has multiple jobs $J_u$, $J := \cup_{u \in U} J_u$, and each job $j \in J$ has to be scheduled on one of multiple machines $M$. For the experimental evaluation, we only use instances with a single machine, i.e. $|M| = 1$, but to be consistent with the literature, we formulate our methods for an arbitrary number of machines. Each job $j \in J$ has a duration $d_j \in \mathbb{N}$ in terms of the number of discrete timesteps. Refer to the set of timesteps in which job $j$ would run if started in timestep $t$ with $T_j[t] = \{t, t+1, \ldots, t+d_j-1\}$ and denote with $T_j^{\text{job}} \subseteq T$ the set of candidate starting times of job $j \in J$, restricted due to the fact that a job has to finish until the end of a day and is not allowed to span multiple days. The objective is to minimize time- and machine-dependent costs $c_{it}$ for using

machine $i \in M$ in timestep $t \in T$. It is possible to not schedule a job $j \in J$, and this induces a penalty cost $q_j$.

This scheduling problem can be formulated by the following Integer Linear Program ILP($\mathcal{T}$), which is parameterized with a set of allowed starting times $\mathcal{T}_j$ for each job $j \in J$.

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

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

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

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

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

The binary variables $x_{jit}$ indicate with value one that job $j$ starts on machine $i$ at timestep $t$ and constraints (2)–(4) make sure that each job is scheduled at most once, that jobs do not overlap on the same machine and that jobs of the same user do not overlap.

Running times of jobs $J_u$ also have to take into account user $u$'s availabilities, which are only known partially. To improve the schedule, a limited amount of interaction with the users is allowed. Initially, each user specifies a starting time for each of their jobs that would work for them. In each of multiple rounds, the approach determines a set of most meaningful queries and the users reply to those queries, enriching the knowledge about their availabilities. We denote with $T_u^{\text{avail}}$ the set of timesteps for which user $u$ is known to be available at the current stage of interaction. We consider two different types of queries. The first one has already been used in the literature; it consists of a time interval and the user either accepts it, if they are available in the whole interval, or rejects it otherwise. The second and novel query type consists of a larger timeframe and a duration and the user selects a time interval within the timeframe of the given duration in which they are available, or indicates that there is no such interval within the timeframe. The timeframe is chosen from a predefined set $F$ of timeframes. In our experiments it consists of two timeframes for each day, one from 6am to 2pm, the other from 2pm to 10pm.

## 3 User Models

For high quality queries, a user response that increases the knowledge about relevant user availability intervals has to be likely and to assess the probability distribution over user responses, we use and compare different probabilistic models for the users from the literature [6]. To minimize assumptions about users,

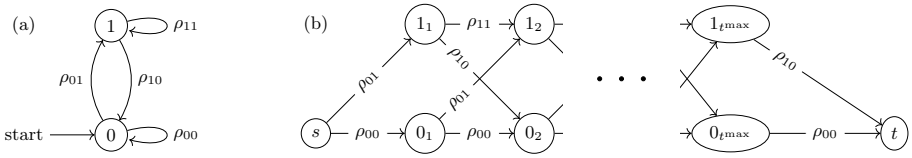


**Fig. 1.** (a) Two-state Markov process and (b) corresponding unrolled state graph. Adapted from [6].

we assume for the first model that the availability of a user in a timestep only depends on the availability of the user in the immediately preceeding timestep. This results in a Markov chain with two states 0 and 1, representing the user being not available and being available, respectively, see also Figure 1a. Before the first timestep, the user is unavailable and from one timestep to the next the user gets available or stays unavailable with probabilities $\rho_{01}$ and $\rho_{00} = 1 - \rho_{01}$, respectively, if they were unavailable, or the user gets unavailable or stays available with probabilities $\rho_{10}$ and $\rho_{11} = 1 - \rho_{10}$, respectively, if they were available. We will refer to this model with MARKOV. The second model assumes the user to be available in up to two intervals within a day. Both intervals are included independently with a given probability and their starting time and duration follow a normal distribution, rounded to the nearest timestep. If the intervals overlap, we take their union. This model is referred to as ADVANCED and can be formulated as a Markov chain as well, see [6] for details.

The models are used to approximate acceptance probabilities of queries and to generate plausible sets of user availabilities. This is done by unrolling the Markov chain associated with the model by duplicating each state for each timestep and adding a source node $s$ and a target node $t$, see Figure 1b for the unrolled state graph for the two-state Markov chain. Each state is associated with the user being unavailable or the user being available, and thus each path corresponds to one set of user availabilities and the product of edge weights results in the probability of that outcome. The knowledge about a day, obtained from user replies, is incorporated by manipulating the graph appropriately to rule out incompatible paths. Acceptance probabilities of queries are computed based on the probability $p^{\text{path}}_{s,v}$ to reach node $v$ from the source node $s$ and on the probability $p^{\text{path}}_{v,t}$ to reach the target node $t$ from node $v$, see [6] for more details on the graph construction, the calculation of $p^{\text{path}}$, and the calculation of query acceptance probabilities from $p^{\text{path}}$. To generate sets of availabilities according to the model and considering the knowledge gathered from user replies, we start in the source node $s$ and iteratively select the next node $n$ proportional to the product of the edge weight and $p^{\text{path}}_{n,t}$.

## 4 Solution Approaches

The task in each round is to find most meaningful queries with high potential to improve the current schedule. This task has been solved in the literature

by first filtering all queries with an estimated acceptance probability below a threshold $p^{\text{lim}}$ from a set of candidate queries and then selecting, by solving an ILP, those queries that improve the objective the most, assuming the queries are accepted [6]. We refer to this approach as MARKOV($p^{\text{lim}}$) or ADVANCED($p^{\text{lim}}$), depending on the underlying user model.

The drawbacks of the probability threshold approach are that queries with probability above the threshold are treated equally, independent of their calculated probabilities, and that correlations between the replies of different queries are not taken into account. Furthermore, the approach is tailored for yes/no queries and does not work for the newly considered timeframe queries. To overcome these deficiencies, we propose the following two-stage stochastic programming approach that identifies queries that minimize expected costs after user replies. This approach is greedy in the sense that it selects those queries in each round that reduce the expected objective value the most.

$$\min \quad \mathbb{E}_{T^{\text{avail}*}}(\text{ILP}(\mathcal{T}(T^{\text{avail}*}, s^{\text{time}}, s^{\text{frame}}))) \tag{6}$$

$$\text{s.t.} \quad \sum_{j \in J} \left( \sum_{t \in T} s^{\text{time}}_{jt} + \sum_{f \in F} s^{\text{frame}}_{jf} \right) \leq b \tag{7}$$

$$s^{\text{time}}_{jt} \in \{0,1\} \qquad j \in J,\ t \in T \tag{8}$$

$$s^{\text{frame}}_{jf} \in \{0,1\} \qquad j \in J,\ f \in F \tag{9}$$

The binary variables $s^{\text{time}}$ and $s^{\text{frame}}$ select the yes/no and timeframe queries, respectively, that will be relayed to the users and (7) limits their number to $b$. Objective (6) is to minimize expected costs for the selected queries, subject to user availabilities $T^{\text{avail}*}$ that are assumed to be distributed according to the user model at hand. The term $\mathcal{T}(T^{\text{avail}*}, s^{\text{time}}, s^{\text{frame}})$ denotes the set of timesteps in which jobs are allowed to start. This set is based on the user replies to the selected queries, which are determined based on the users availabilities $T^{\text{avail}*}$.

As the expected costs involve a hard optimization problem, it is challenging if not impossible to model the exact stochastic program by means of an ILP, and therefore, we obtain queries by solving the sample average approximation [3]. To do so, we first sample $n^{\text{samples}}$ possible sets of user availabilities for each user based on the user model and compute the user reply for each sample and possible query. In case of the first form of yes/no queries, this results in the set of starting times $T_j^{\text{pos,(k)}} \subseteq T$ for each job $j \in J$, for which the reply would be positive for sample $k$. For timeframe queries, this results in a set of timeframes $F_j^{\text{pos,(k)}} \subseteq F$ for each job $j \in J$ in which the reply would be positive and for each of these frames $f \in F_j^{\text{pos,(k)}}$ the starting time $t_{jf}^{\text{reply,(k)}} \in T$ that the user would choose randomly from their possible starting times within the timeframe.

With that terminology, the ILP for the approximation of the stochastic program can be formulated as $n^{\text{samples}}$ instances of the ILP from [6] of the core scheduling problem without restricting job starting times yet, combining the objectives by taking the mean. This ILP uses variables $x^{(k)}_{jit} \in \{0,1\}, j \in J, i \in$

$M, t \in T$ for each sample $k$ to indicate with value one that job $j$ is scheduled on machine $i$ with starting time $t$. We restrict job starting times according to the selected queries and the corresponding user replies by adding the following constraints for each sample $k$:

$$x_{jit}^{(k)} = 0 \qquad j \in J, i \in M, t \in T \setminus T_j^{\mathrm{pos,(k)}} \quad (10)$$

$$\sum_{j' \in J_u} \sum_{t' \in T | T_j[t] \subseteq T_{j'}[t']} \left( s_{j't'}^{\mathrm{time}} + \sum_{f \in F_{j'}^{\mathrm{pos,(k)}} | t_{j'f}^{\mathrm{reply,(k)}} = t'} s_{j'f}^{\mathrm{frame}} \right) \geq$$
$$\sum_{i \in M} \sum_{j' \in J_u} \sum_{t' \in T | T_j[t] \subseteq T_{j'}[t']} x_{j'it'}^{(k)} \qquad u \in U, j \in J_u, t \in T, T_j[t] \not\subseteq T_u^{\mathrm{avail}} \quad (11)$$

Equation (10) prevents starting times that would be rejected by the user. Note that $T_j^{\mathrm{pos,(k)}}$ also includes starting times for which we already know that the user would accept them. Inequality (11) only allows a starting time that we are not sure about yet if a query is selected whose time interval covers the jobs' execution time. The sums over $j'$ and $t'$ on the left side account for the fact that, given an accepted query of a longer job, also a shorter job could be scheduled within the time interval of the query, and the sums over $j'$ and $t'$ on the right side strengthen the formulation based on the idea that a disallowed shorter interval also disallows a longer running job when covering the shorter interval. Note that these constraints are slightly more restrictive than they need to be to make the formulation easier and more practical. For instance, if the time intervals of two selected queries overlap, jobs could be scheduled in the union of those intervals and this is not covered by the formulation above.

## 5 Experimental Results

To test our approach, we use the smallest instances[3] of [6] with one machine, six users and 24 jobs since the sample average approximation of the stochastic program does not scale to larger instances when using a reasonably large number of samples. The time horizon of those instances consists of five days, each day starting at 6am and ending at 10pm with four discrete timesteps per hour. Users are simulated and their availability times, which are not (completely) known to the scheduler, are generated according to the ADVANCED model for which we use the following values. The probability of including an interval is 90% and the normal distributions have a standard deviation of one hour. The mean starting times of the intervals are 9am and 1pm, respectively, and their mean durations are four and five hours, respectively. Based on their generated availability times, the users select one possible starting time for each of their jobs uniformly at random and the hypothetical running times of those jobs form the initial knowledge about the user availabilities.

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

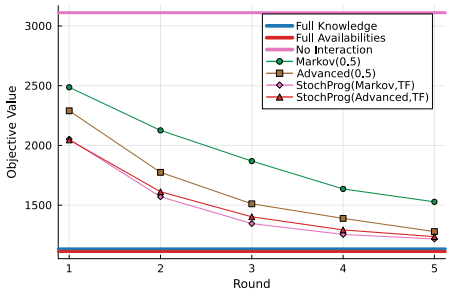


**Fig. 2.** Convergence comparison of the threshold approach with the stochastic programming approach.

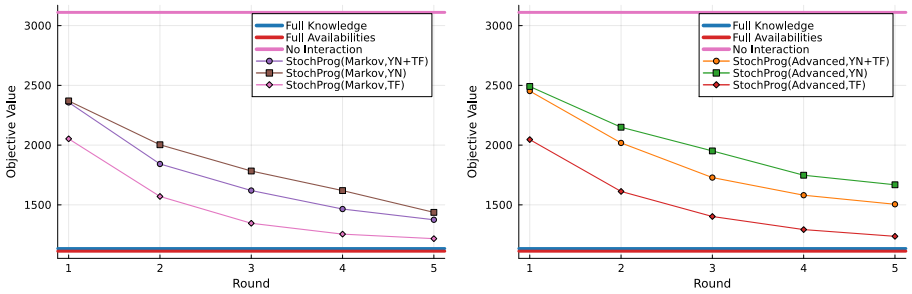


**Fig. 3.** Convergence comparison for the stochastic programming approach when allowing different types of queries for the Markov user model (left) and the advanced user model (right).

We implemented the approaches in Julia 1.10 and used the solver Gurobi 11.0 to solve the ILPs. Each experiment was performed on a single thread of an Intel Xeon E5-2640 v4 with a timelimit of 120 minutes for each ILP. For the sample average approximation of the stochastic program, we use $n^{\text{samples}} = 50$ samples. The optimality gaps of all solutions found by Gurobi are below 45% and their median is 4.3%.

Figures 2 and 3 compare the performance of the different approaches with different parameters. They plot the average objective value over 30 instances of the best schedule that is possible with a certain knowledge about user availabilities. The horizontal lines show the objective value with the knowledge that is available before any user interaction ("No Interaction"), assuming full knowledge about user availabilities ("Full Knowledge"), and assuming that all users are—hypothetically—available all the time ("Full Availabilities"). The curves show the average objective values of the best schedule with the knowledge gathered by one of the approaches with one to five interaction rounds.

Figure 2 shows a comparison of the stochastic programming approach StochProg(Markov, TF) and StochProg(Advanced, TF) with the probability

threshold approach MARKOV(0.5) and ADVANCED(0.5) from [6]. The best configurations of the approaches with $p^{\text{lim}} = 0.5$ and only timeframe queries, respectively, were chosen. The stochastic programming approach clearly performs better with a gap to the full knowledge case after five rounds of 7.3% and 9% for the MARKOV and ADVANCED user models, respectively, compared to gaps of 34.7% and 12.7%. Apparently, both user models work almost equally well within the stochastic programming approach, when using only timeframe queries.

Figure 3 compares the stochastic programming approach when allowing only yes/no queries ("YN"), only timeframe queries ("TF") and both ("YN+TF") for the MARKOV model and for the ADVANCED model. The approach with timeframe queries performs with final gaps of 7.3% (MARKOV) and 9% (ADVANCED) significantly better than the one with yes/no queries with final gaps of 26.7% (MARKOV) and 47% (ADVANCED). This is not surprising since the user gives more information with each reply to a query. A bit surprising is the fact that only allowing timeframe queries results in a better performance than allowing both types of queries (final gaps 21.2% and 32.7%) considering the fact that the latter could only select timeframe queries. We explain this with the approximation error related to the limited amount of samples. While the expected objective predicts the actual objective after the users replies quite well for timeframe queries with a mean difference of 1.1%, allowing only yes/no queries results in a mean difference of 17.7%. This indicates that timeframe queries are better compatible with the sample average approximation than yes/no queries. But yes/no queries might still benefit from stochastic programming, either by using more samples for the approximation, or by using a more advanced approach to solve it such as the one described in [4], where the expected objective is estimated by a neural network.

## 6 Conclusion

We investigated the problem of selecting meaningful queries of two different types in an interactive scheduling setting, proposed a stochastic programming approach and compared it with the state-of-the-art. The stochastic programming approach converges significantly faster than the probability threshold approach from the literature. Furthermore, the query type that asks the user to specify alternative running times for their jobs works better than proposing alternative running times to the users.

To solve the stochastic program, we formulated the sample average approximation and solved it with an Integer Linear Programming solver. While this is easy to implement and works reasonably well, it does not scale well to larger instances and has a considerable approximation error. In future work, we plan to overcome these issues by using a more advanced stochastic programming method, such as estimating expected costs via a neural network.

## References

1. Anghinolfi, D., Paolucci, M., Ronco, R.: A bi-objective heuristic approach for green identical parallel machine scheduling. European Journal of Operational Research **289**(2), 416–434 (2021)
2. Jatschka, T., Raidl, G.R., Rodemann, T.: A general cooperative optimization approach for distributing service points in mobility applications. Algorithms **14**(8) (2021)
3. Kleywegt, A.J., Shapiro, A., Homem-de Mello, T.: The sample average approximation method for stochastic discrete optimization. SIAM Journal on optimization **12**(2), 479–502 (2002)
4. Kronqvist, J., Li, B., Rolfes, J., Zhao, S.: Alternating mixed-integer programming and neural network training for approximating stochastic two-stage problems. In: Nicosia, G., Ojha, V., La Malfa, E., La Malfa, G., Pardalos, P.M., Umeton, R. (eds.) Machine Learning, Optimization, and Data Science. pp. 124–139. Springer Nature Switzerland, Cham (2024)
5. Varga, J., Raidl, G.R., Rönnberg, E., Rodemann, T.: Interactive job scheduling with partially known personnel availabilities. In: Dorronsoro, et al. (eds.) OLA 2023: Optimization and Learning. Communications in Computer and Information Science, vol. 1824, pp. 236–247. Springer (2023)
6. Varga, J., Raidl, G.R., Rönnberg, E., Rodemann, T.: Scheduling jobs using queries to interactively learn human availability times. Computers & Operations Research **167**, 106648 (2024)
7. 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)
8. Wollstadt, P., Krüger, M., Wiebel-Herboth, C.B.: Quantifying cooperation between rule-based hanabi agents using information theory. In: Lukowicz, P., Mayer, S., Koch, J., Shawe-Taylor, J., Tiddi, I. (eds.) HHAI 2023: Augmenting Human Intellect: Proceedings of the Second International Conference on Hybrid Human-Artificial Intelligence, Frontiers in Artificial Intelligence and Applications, vol. 368, pp. 422–425. IOS Press (2023)