Query Timing in Interactive Job Scheduling

Johannes Varga[1], Harald Korinek[1], Günther R. Raidl[1], and Tobias Rodeman[2]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{jvarga,raidl}@ac.tuwien.ac.at, haraldkor@protonmail.com
[2] Honda Research Institute Europe, Offenbach, Germany
tobias.rodemann@honda-ri.de

Abstract. We consider the Interactive Job Scheduling Problem, where human users want to schedule jobs on a machine, but are also restricted in the times in which they are available to operate the machine. Querying the relevant parts of these availabilities interactively as needed by the optimization is an non-intrusive way to come up with a schedule that incorporates the preferences of the users. Naturally, the reply time of users can differ significantly. In this work, we propose and investigate two new approaches to determine at which times to compute and send new queries, improving upon an approach from the literature, where time is potentially wasted by waiting for all replies before computing new queries. One of our approaches does not wait whenever receiving a reply, while the other one waits for a fixed amount of time. Results demonstrate that especially if the amount of available time is low, our approaches significantly outperform the approach from the literature.

1 Introduction

We consider the Interactive Job Scheduling Problem in which human users, e.g. patients in radiation therapy [16, 1], regularly require access to a limited and high-demand resource, e.g. a linear accelerator for the treatment of cancer. To avoid no-shows and frustration of the users, it is important to consider the preferences of the users, in particular the times in which they are available to use the resource. Usually, it is not viable to ask the users to specify their full availabilities in all detail. Instead, we assume that users initially propose only few timeslots in which they are available and then a scheduling system, a kind of conversational recommender system [4], interacts with the users to find out more about the relevant parts of their preferences. During interaction, the scheduling system proposes alternative timeslots to the users, who accept or reject them depending on their availability times. Previous work [15] does this in multiple interaction rounds, and in each round, the scheduling system sends out a small number of queries and awaits all replies before starting the next round. In practice, some queries will be answered immediately, while others remain unanswered for a longer time. Since the scheduling system waits for all of them, each round takes up a significant amount of time, reducing the total amount of queries that can be sent in a fixed timeframe. In general, it is not easy to determine when

to send queries. Waiting for more replies before sending queries provides more knowledge about the users, which results in better queries that deliver more improvement of the schedule quality per query. This comes at the price of the time needed to find a schedule: shorter waiting times yield quicker improvements. In this work, we explore different strategies for the timing of interaction in the Interactive Job Scheduling Problem to achieve better schedules with the same time and interaction budget.

For the durations that the users require for their reply after having received queries, we consider two models: One with normally distributed, the other with exponentially distributed durations. We develop two approaches for determining when to send queries to reduce the overall interaction time that is required to achieve a schedule of good quality. In the first approach, the *responsive approach*, the scheduling system starts by computing and sending out one query per user. Whenever a reply is received, a new query is computed and sent, effectively keeping the number of unanswered queries constant. This approach has the drawback that it does not consider interdependencies between queries: for instance consider two queries that could improve the schedule in conjunction but not individually; since only one query is computed in each step, these might never be considered. Therefore, we refine it to the more advanced *hybrid approach*, which additionally waits for a fixed amount of time between receiving a reply and computing and sending new queries.

We build upon the line of work of Varga et al. [15, 14]. They proposed the Interactive Job Scheduling Problem and developed a scheduling approach to solve it. In their approach, they use two different user models and a Markov-chain based computation of acceptance probabilities for queries. By doing simulation runs they find that the schedule quickly improves with only few queries when using any of the user models within the scheduler. In [13], the authors extend the scenario and scheduler to additionally consider another type of queries. Here, we will specifically look at the *timing* of replies and queries.

A related research field is interactive optimization [7] where the search through a solution space is guided by humans. Usually, only one human guides the search, but some works also consider interaction with a larger user base. For instance, the cooperative optimization approach of Jatschka et al. [5] interacts with human users to find the most suitable placement of mobility service stations, for instance battery swapping stations, in a city. Similarly, in calendar scheduling [8, 3], meetings between people are scheduled by scheduling agents based on input from human users.

Effectively, the scheduler performs active learning [10] on the users by placing queries to learn about the most relevant parts of the users' availabilities. In batched active learning [2, 12], multiple datapoints are selected and evaluated at once and then used for retraining before computing the next set (aka batch) of datapoints. Contrary, in stream-based active learning, datapoints are sent for evaluation one by one and when there is a verification latency [9], the result of the evaluation is delayed. Comparing to our setting, batched active learning is related to the baseline, round-based approach and stream-based active learning

with verification latency corresponds to the extended setting that also considers the timing of the replies and queries.

The remaining parts of this work are organized as follows: We present a formal definition of the scheduling problem in the next section. Section 3 summarizes how the scheduler computes queries. Afterwards, Section 4 shows how users are simulated, and Section 5 discusses the responsive and the hybrid approaches. Section 6 shows an experimental evaluation and Section 7 concludes the work.

2 Interactive Job Scheduling

We denote with U the set of users, with J_u the set of jobs associated with each user $u \in U$ and with $J := \bigcup_{u \in U} J_u$ the set of all jobs. The discretized planning time horizon T consists of $t^{\text{max-day}}$ days, each job $j \in J$ has a duration $d_j \in \mathbb{N}$ and has to be scheduled within T on one of the machines M. Running a job on a machine $i \in M$ at timestep $t \in T$ induces time- and machine-dependent costs c_{it}. Not running a job $j \in J$ causes penalty costs q_j. The objective is to minimize total costs. Let $T_j^{\text{job}} \subseteq T$ be the set of timesteps in which the job j can start, considering that it cannot span multiple days. Furthermore, refer with $T_j[t] = \{t, t+1, \ldots, t+d_j-1\}$ to the timesteps in which job $j \in J$ would run if started at timestep t.

For completeness, the Integer Linear Program from Varga et al. [15] is stated in the following. It is referred to as $ILP(\mathcal{T})$ and parameterized by 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} determine job starting times, i.e. $x_{jit} = 1$ means that job $j \in J$ starts on machine $i \in M$ at timestep $t \in T$ and constraint (2) ensures that each job starts at most once. Inequality (3) enforces that no jobs are executed on the same machine at the same time and constraint (4) guarantees that no jobs of the same user overlap.

User u's scheduled jobs J_u have to respect the times in which user u is available and these times are only partially known to the scheduling system. Initially, users communicate a single possible starting time for each of their jobs. Since this usually leads to a considerable amount of collisions, the scheduling

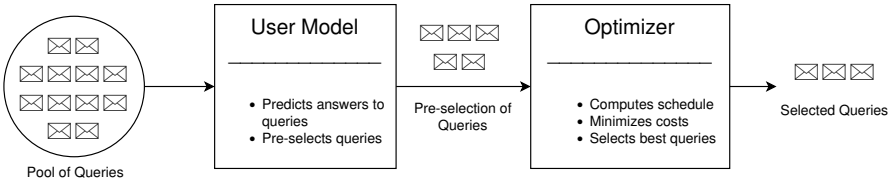

Fig. 1. Scheduling approach.

system then interacts with the users to find out more about their availability times. In each of a small number of rounds, the scheduling system sends queries to users. A query asks a user, whether they are available in a given time interval and is answered by the user with either yes or no. At any stage, we refer with T_u^{avail} to the currently known availability times of user u. The current objective value is then the objective value of the optimal schedule that respects these currently known availability times T_u^{avail}.

3 Scheduling System

In each round, we use the scheduling system proposed by Varga et al. [15], see also Figure 1 for an overview. It considers intervals with job durations placed throughout the time horizon in all possible ways that do not span over multiple days. The queries corresponding to those intervals are evaluated with a user model, resulting in a probability for each query that the user will accept it. The user model assumes that a user's availability only changes a few times throughout a day and it combines this assumption with already obtained replies to previous queries. Only queries that have a predicted chance of being accepted of at least 75% are considered further. The threshold of 75% turned out to perform best in [15]. Those queries, denoted as T_j^{query} for a job $j \in J$, are then fed into the optimizer, which selects a limited amount of $\bar{b}$ queries that, if accepted, minimize costs. This is achieved by solving the ILP$((T_j^{\text{feas}} \cup T_j^{\text{query}})_{j \in J})$ where T_j^{feas} is the set of feasible starting times of job $j \in J$ with respect to the known user availability times and with the additional constraints

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

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

These constraints ensure that at most $\bar{b}$ queries are selected and that queries do not target the same user on the same day twice, since this might lead to higher redundancy in their replies.

4 User Simulation

We assume that users always answer truthfully based on the times in which they are available. In particular, we assume that they do not respond carelessly after some queries and that they do not try to trick the system. A simulated user answers based on their set of availability times. If they are available during the whole time interval of a query, they accept the query, otherwise they it. In our empircial study, availability times of simulated users are derived from the Dutch Time-use-survey [11]. For more details see Varga et al. [15].

In former work on the Interactive Job Scheduling Problem, the time that users need to answer was not considered and thus assumed to be short enough for practical applications. As all replies are collected before computing a new schedule and starting a new round, individual users that require a long time for their reply may substantially delay the whole process. Therefore, in this work we consider the reply time dynamics of users and their impacts. We use an idealized model, where the reply delay of each query is represented by a random variable, and these variables are identically distributed and independent. More specifically, we consider two different distributions: The normal distribution and the exponential distribution [6]. In the literature, also ExNormal distributions are often used [17], which combine normal and exponential distributions, and for which we may expect results lying inbetween those of ours.

5 Approaches for the Timing of Queries

Blocking approach. As baseline to compare our approaches to, we use the blocking approach from [15], also referred to as round-based interaction. Figure 2 shows the timeline. Initially, users communicate, which jobs they want to run, i.e., how often and how long they want to use the machine, and propose one possible timeslot for each of their jobs. The remaining interaction is done in multiple rounds. In each round, the scheduling system computes and sends out b queries, which are answered by the users according to the user model from Section 4. The value of b is given as a parameter to the approach and is fixed for the whole procedure. The scheduling system waits for all replies before continuing with computing the queries of the next round. The main disadvantage of the blocking approach is that waiting for replies takes time and a single long reply delay can significantly increase the time that is needed to come up with a schedule. To mitigate this drawback, we propose two approaches: the responsive approach and the hybrid approach.

Responsive Approach. A pseudocode for the algorithm of the responsive approach is shown in Algorithm 1 and Figure 3 depicts an exemplary timeline. As in the blocking approach, the scheduling system starts by computing and sending b queries after receiving jobs and proposed time intervals by the users. Whenever it gets a reply of a user, it immediately computes and sends a new query. Note that this new query does not have to go to the same user who just

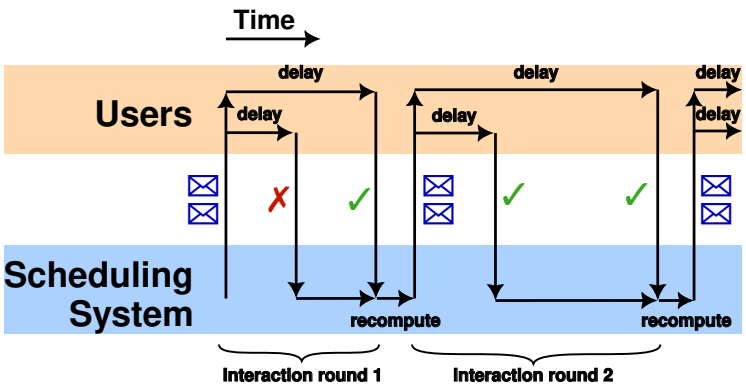

Fig. 2. Blocking approach with round-based interaction.

Algorithm 1: Responsive approach

```
Send b queries
while Time remaining do
    Wait for reply
    Send one query
Compute final schedule
```

Algorithm 2: Hybrid approach

```
Send b queries
while Time remaining do
    Wait for reply
    Wait for t^wait time units
    Send one query for each received reply
Compute final schedule
```

replied. This is repeated as long as time remains. Queries are computed with the system explained in Section 3, setting parameter $\bar{b}$ to one except for the first queries for which it is set to b. If a query has not been answered yet, it prevents that a new query to the same user overlaps with it. This is realised by adding the constraint

$$\sum_{\substack{t^* \in \mathcal{T}_j \\ t - d_j + 1 \leq t^* \leq t'}} x_{jit^*} = 0 \qquad i \in M, (u, [t, t']) \in Q, j \in J_u \tag{8}$$

where all open queries $(u, [t, t'])$ are collected in the set Q, consisting of a time interval $[t, t']$ and the user u that the query has been sent to.

Hybrid Approach. Our second approach is a combination of the blocking and the responsive approach. While the blocking approach awaits all replies and the responsive does not wait at all when receiving a query, the hybrid approach waits for a fixed time, which is given as the parameter t^{wait}. Algorithm 2 shows the procedure and Figure 4 provides an exemplary interaction timeline. Again, the approach starts by computing and sending b queries. After it receives the first reply, it waits for t^{wait} time units during which it might receive further replies. The approach then computes and sends as many queries as it received replies. This cycle is repeated as long as time remains. The computation of queries is

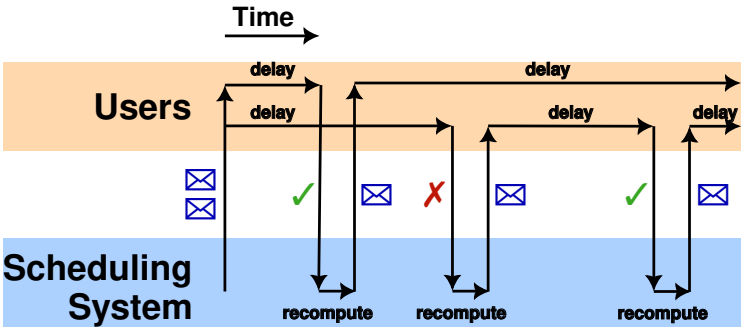

Fig. 3. Responsive approach.

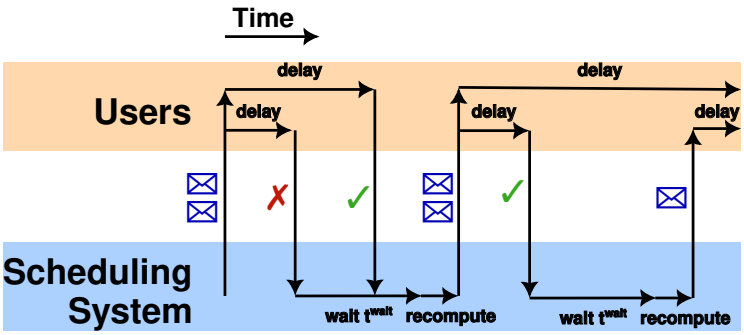

Fig. 4. Hybrid approach.

done in the same way as for the responsive approach, but using appropriate values for $\bar{b}$.

6 Experimental Results

We evaluate the approaches with instances of [15][3]. We select the 30 instances with two machines, twelve users and 48 jobs since solving times of the optimizer are relatively low, still they bring the complexity of considering multiple machines. Experiments were performed on an AMD EPYC 9554P processor, using Julia 1.12 and Gurobi 13 for the optimization core. The reply delays of the users are distributed according to an exponential distribution with mean 100 and a normal distribution with mean of 100 and standard deviation of 50 time units, respectively. For the blocking and responsive approach and each parameter value $b \in \{1, 2, 3, 6, 9, 12, 18, 24, 36, 48\}$, one interaction simulation of at least 600 time units and at least 60 received replies to queries was performed. Similarly, for the hybrid approach and each parameter value combination of

[3] The instances and the description of their format are available at https://www.ac.tuwien.ac.at/research/problem-instances/#ijsp

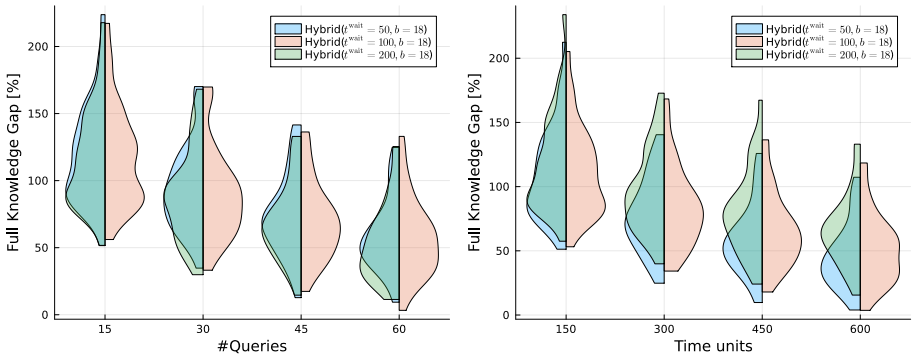

Fig. 5. Comparison between different values of t^{wait} for the hybrid approach with $b = 18$ and the exponential reply delay.

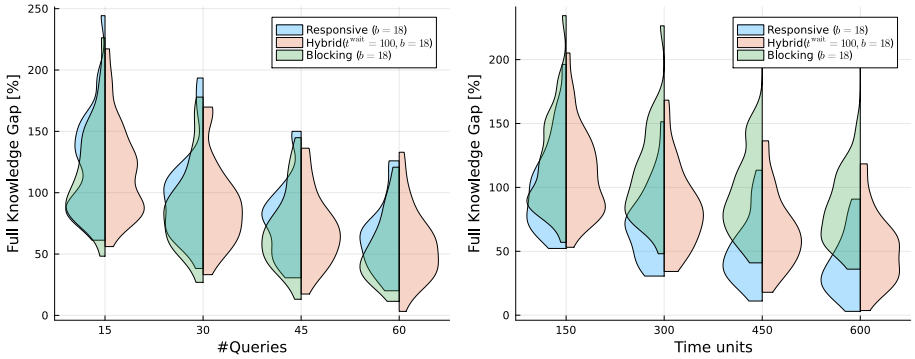

Fig. 6. Comparison between the blocking, responsive and hybrid approach with exponential reply delay and best working parameter values ($b = 18$, $t^{\text{wait}} = 100$).

$b \in \{1, 2, 3, 6, 9, 12, 18, 24, 36, 48\}$ and $t^{\text{wait}} \in \{50, 100, 200\}$ one interaction simulation was performed. Lower time or interaction budgets can be reasonable as well and therefore, we also record intermediate states and use them for the evaluation.

Figure 5 compares the different values of the parameter t^{wait} for the hybrid approach and the exponential reply delay distributions. The plots show the development of the full knowledge gap over the number of received queries and time, respectively. The full knowledge gap is the relative gap between the schedule quality with current knowledge about the users' availabilities and the optimal schedule quality with full knowledge about the users' availabilities. At four points on the x-axis, the distribution over the runs with all 30 instances is plotted. As can be seen, the differences between the different values for t^{wait} are neglectible and therefore, we use a value of $t^{\text{wait}} = 100$ for the next evaluations.

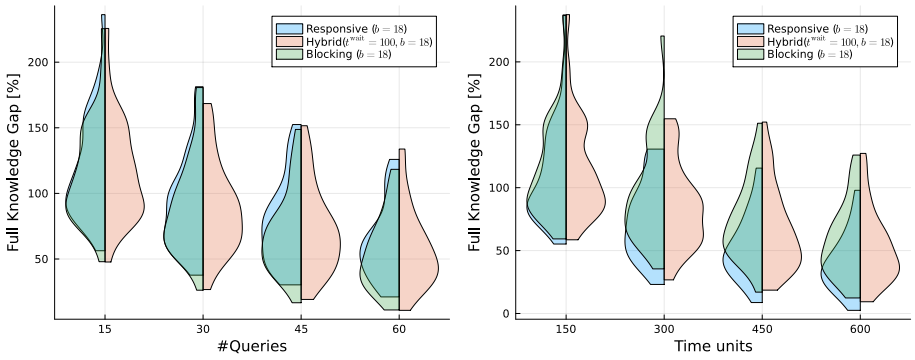

Fig. 7. Comparison between the blocking, responsive and hybrid approach with normal reply delay and best working parameter values ($b = 18$, $t^{\text{wait}} = 100$).

Figure 6 shows a comparison between the three approaches with the exponential reply time distribution with parameter values that proved to work well. For a fixed time budget, the responsive approach works best, clearly outperforming the blocking approach. On the contrary, the responsive approach performs slightly worse when given a fixed interaction budget, which is expected, since it works with less information when computing new queries. The hybrid approach works similarly well as the responsive approach.

Figure 7 makes the same comparison for the normal reply time distribution. Here, the responsive approach works slightly better than the other approaches for fixed time budget and slightly worse for fixed interaction budget. Overall, the insights are similar as for the exponential distribution, except that the differences are smaller. The reason for the larger differences is that the long tail of the exponential distribution makes the blocking approach perform worse, since it waits for all replies in each round. The normal distribution does not have a long tail and therefore shows smaller differences. We expect that the results with the exponential distribution also transfer to the ExNormal distribution, which is often used to approximate human reply times [17]. In the further evaluation, we therefore also only focus on the more interesting exponential distribution.

Figure 8 shows the payoff between between schedule quality and time for a fixed interaction budget of 60 queries. Each point in the plot represents a run with one approach using a specific parameter value combination. For small time budgets, the responsive approach works best and for high time budgets, the blocking approach provides the best schedules. In between and for the majority of time budgets, the hybrid approach outperforms the other two approaches. This is expected behavior: While the strength of the responsive approach is speed and of the blocking approach is schedule quality, the hybrid approach combines the strengths of the other two approaches.

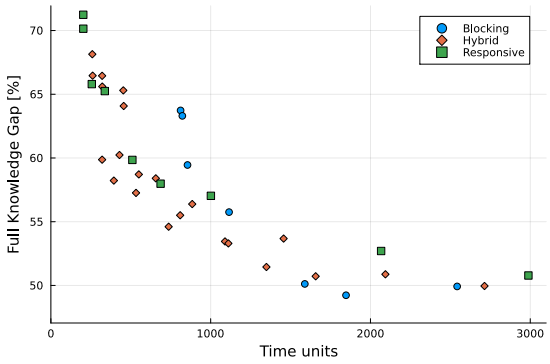

Fig. 8. Payoff between schedule quality and time for the blocking, hybrid and responsive approach.

7 Conclusion

We considered a job scheduling framework in which users want to perform jobs, but are limited in their temporal availabilities. To minimize user burden but allow for high quality schedules at the same time, user availabilities are determined through interactive queries. In this setting, we investigated reply time dynamics and extended a user simulation from the literature by a time delay of the replies to queries. To improve on the time that is required to come up with a good schedule, we propose two approaches to decide when to send queries. While the responsive approach makes sure that the number of open queries stays constant, the hybrid approach allows small deviations. Results show that for a fixed interaction budget the responsive approach delivers good results when the overall scheduling time is rather short, while the hybrid approach exhibits advantages for longer scheduling times. Future work might consider ExNormal distributions and different distributions for different users for a more realistic model of the reply time dynamics.

References

1. Ahmadi-Javid, A., Jalali, Z., Klassen, K.J.: Outpatient appointment systems in healthcare: A review of optimization studies. European Journal of Operational Research **258**(1), 3–34 (2017)
2. Alvi, A., Ru, B., Calliess, J.P., Roberts, S., Osborne, M.A.: Asynchronous batch Bayesian optimisation with improved local penalisation. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 253–262. PMLR (2019)
3. Gervasio, M.T., Moffitt, M.D., Pollack, M.E., Taylor, J.M., Uribe, T.E.: Active preference learning for personalized calendar scheduling assistance. In: Proceedings of the 10th international conference on Intelligent user interfaces. pp. 90–97. IUI '05, Association for Computing Machinery (2005)

4. Jannach, D., Manzoor, A., Cai, W., Chen, L.: A survey on conversational recommender systems. ACM Comput. Surv. **54**(5) (2021)
5. Jatschka, T., Raidl, G.R., Rodemann, T.: A general cooperative optimization approach for distributing service points in mobility applications. Algorithms **14**(8) (2021)
6. Mahmud, J., Chen, J., Nichols, J.: When will you answer this? estimating response time in twitter. In: Proceedings of the International AAAI Conference on Web and Social Media. vol. 7, pp. 697–700 (2013)
7. Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N.: A review and taxonomy of interactive optimization methods in operations research. ACM Transactions on Interactive Intelligent Systems (TiiS) **5**(3), 1–43 (2015)
8. Oh, J., Smith, S.F.: Learning user preferences in distributed calendar scheduling. In: International Conference on the Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science, vol. 3616, pp. 3–16. Springer (2004)
9. Pham, T., Kottke, D., Krempl, G., Sick, B.: Stream-based active learning for sliding windows under the influence of verification latency. Machine Learning **111**(6), 2011–2036 (2022)
10. Settles, B.: Active learning literature survey (2009)
11. Sociaal en Cultureel Planbureau: Tijdsbestedingsonderzoek 2005 - TBO 2005 (2005). https://doi.org/10.17026/dans-znn-5xvz, https://doi.org/10.17026/dans-znn-5xvz
12. Tran, A., Eldred, M., Wildey, T., McCann, S., Sun, J., Visintainer, R.J.: aphbo-2gp-3b: a budgeted asynchronous parallel multi-acquisition functions for constrained bayesian optimization on high-performing computing architecture. Struct. Multidiscip. Optim. **65**(4) (2022)
13. Varga, J., Raidl, G.R., Rodemann, T.: Selecting user queries in interactive job scheduling. In: International Conference on Computer Aided Systems Theory. Lecture Notes in Computer Science, vol. 15172, pp. 202–210. Springer (2024)
14. 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)
15. 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)
16. Vieira, B., Demirtas, D., van de Kamer, J.B., Hans, E.W., Jongste, W., van Harten, W.: Radiotherapy treatment scheduling: Implementing operations research into clinical practice. Plos one **16**(2), e0247428 (2021)
17. White, S.R., Preston, M.W., Swanson, K., Laubach, M.: Learning to choose: behavioral dynamics underlying the initial acquisition of decision-making. Eneuro **11**(5) (2024)