Preference Elicitation with Interdependency and User Bother Cost

Tiep Le
New Mexico State University
Las Cruces, NM, USA
tile@cs.nmsu.edu

Atena M. Tabakhi
Washington University in St. Louis
St. Louis, MO, USA
amtabakhi@wustl.edu

Long Tran-Thanh
University of Southampton
Southampton, UK
l.tran-thanh@soton.ac.uk

William Yeoh
Washington University in St. Louis
St. Louis, MO, USA
wyeoh@wustl.edu

Tran Cao Son
New Mexico State University
Las Cruces, NM, USA
tson@cs.nmsu.edu

ABSTRACT

Agent-based scheduling systems, such as automated systems that schedule meetings for users and systems that schedule smart devices in smart homes, require the elicitation of user preferences in oder to operate in a manner that is consistent with user expectations. Unfortunately, interactions between such systems and users can be limited as human users prefer to not be overly bothered by such systems. As such, a key challenge is for the system to efficiently elicit key preferences without bothering the users too much.

To tackle this problem, we propose a cost model that captures the cognitive or *bother cost* associated with asking a question. We incorporate this model into our iPLEASE system, an interactive preference elicitation approach. iPLEASE represents a user's preferences as a matrix, called *preference matrix*, and uses heuristics to select, from a given set of questions, an efficient sequence of questions to ask the user such that the total bother cost incurred to the user does not exceed a given bother cost budget. The user's response to those questions will partially populate the preference matrix. It then performs an exact matrix completion via convex optimization to approximate the remaining preferences that are not directly elicited. We empirically apply iPLEASE on randomly-generated problems as well as on a real-world dataset for the smart device scheduling problem to demonstrate that our approach outperforms other non-trivial benchmarks in eliciting user preferences.

KEYWORDS

Preference Elicitation; Matrix Completion; User Bother Cost

ACM Reference Format:
Tiep Le, Atena M. Tabakhi, Long Tran-Thanh, William Yeoh, and Tran Cao Son. 2018. Preference Elicitation with Interdependency and User Bother Cost. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 10 pages.

1 INTRODUCTION

Multi-agent researchers have proposed agent-based systems to solve a number of *scheduling problems*, including the scheduling of meetings [26, 33, 36] and the scheduling of smart devices in smart homes [14, 32]. Such scheduling problems have the unique property that its utility function is highly dependent on users, that is, the agents responsible for performing the scheduling task need to know the preferences of its users to ensure that the proposed schedule is consistent with user expectations. For example, in a meeting scheduling problem, the agent will need to know if users have specific time periods for which no meetings should be scheduled as well as preferred time periods for when certain types of meetings should be scheduled. We call this class of scheduling problems *preference-dependent scheduling* (PDS) problems. The objective in PDS problems is to find a schedule that is optimized for a given set of user preferences. This characteristic is in contrast to other scheduling problems, such as job-shop scheduling problems [3], where the goal is to find a schedule that minimizes the makespan.

Unfortunately, PDS systems cannot have an unlimited amount of interactions with users to elicit *all* preferences; human users are likely to be bothered by the questions asked and are willing to only answer few questions. As such, a key challenge in this area of *preference elicitation* is the identification of the limited set of questions to ask users in order to obtain as much useful information as possible. Most existing methods have thus far made the assumption that all possible questions asked are equally bothersome or, in other words, they all have the same bother costs (e.g., [35]). As this assumption is not likely to hold in practice – users are likely to be less bothered by a simple yes/no question compared to an open-ended one – we incorporate a cost model that models the cognitive or bother cost associated with asking a question. We relate this bother cost to the amount of information obtained through the answers provided by users. Then, we seek to identify the best set of questions to ask such that its respective total bother cost is within a user-defined bother cost budget.

Additionally, we assume that in PDS problems, there are strong inter-dependencies between the preferences. For example, in meeting scheduling problems, the preferences for a meeting with one colleague are likely very similar to the preferences for a meeting with a different colleague. Similarly, in smart device scheduling problems, the preferences for scheduling a smart light is likely very similar to the preferences for other lights in the same room. To exploit this assumption, we first represent the preferences in a *preference matrix*, which is a matrix whose rows correspond to the tasks that need to be scheduled (e.g., meetings in meeting scheduling

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

problems or smart devices in smart device scheduling problems) and columns correspond to time steps for when they can be scheduled. Then, we perform *exact matrix completion* via convex optimization [6] to approximate the remaining preferences in the matrix that are not directly elicited.[1] At a high level, exact matrix completion attempts at minimizing the nuclear norm of the preference matrix under the assumption that rank of the matrix is small.

We combine both orthogonal contributions – bother costs and matrix completion that exploit inter-dependencies between preferences – into a single integrated framework, which we call iPLEASE (interactive preference learning and elicitation without annoying users). At a high level, iPLEASE will ask questions to elicit preferences, each of which fills some elements in the preference matrix, and approximate the remaining elements using matrix completion techniques [6]. Its goal is to identify a set of questions to ask users such that the preference matrix can be approximated as accurately as possible and that the total bother cost incurred by all the questions asked does not exceed a user-defined bother cost budget. Towards this end, iPLEASE formulates this resource-constrained optimization problem as a variant of the knapsack problem, and uses novel heuristics to identify the set of questions to ask users.

Application to the Smart Home Domain: While iPLEASE can be used in the elicitation of preferences for any PDS problems, we use the *demand-side management* (DSM) of smart homes as an example application in this paper. In this application, autonomous software agents are deployed in smart plugs in homes, acting on behalf of the homeowners to remotely switch devices (e.g., lights and washing machines) on and off. By considering the cost of electricity, which differs throughout the day, each agent can schedule the device usage during off-peak hours, when electricity is cheaper.

The aim in most state-of-the-art DSM approaches is to find a solution (i.e., schedule for all devices) that maximizes monetary savings to the homeowners as well as the comfort level of homeowners [29, 43]. A key limitation of these techniques is that they typically ignore both the bother cost and the inter-dependencies between preferences (see Related Work for more details) during their elicitation phase. Thus, with iPLEASE we propose the first DSM system that can efficiently manage the energy consumption of homeowners while taking into account their bother costs and the interdependencies between preferences inferred.

Our Contributions:

- We introduce iPLEASE, the first preference elicitation approach that takes into account user bother costs as well as exploit the inter-dependencies between preferences during the questioning process.
- We propose two heuristics for choosing questions to ask: (1) A greedy myopic heuristic that repeatedly chooses the question that provides the largest utility given the current partially-filled preference matrix, subject to the constraint that the bother cost of the question is no larger than the remaining bother cost budget; and (2) A more holistic approach that chooses a subset of questions that provides the largest utility given the initial partially-filled preference matrix, subject to the sum of the bother cost of the questions is no larger than the total budget.
- We compare these heuristics on random and a well-known real-world dataset from the DSM domain, and show that they outperform non-trivial benchmarks in preference elicitation.

[1] Our approach can be easily extended to *tensors*, which can capture more complex user preference structures, by replacing matrix completion methods with tensor completion algorithms. However, for the sake of simplicity, we only consider preference matrices in this paper.

2 BACKGROUND

We review the topic of exact matrix completion via convex optimization [6], which will be used to recover approximately the complete preference matrix.

Let $M \in \mathbb{R}^{m \times n}$ be a matrix. $M_{i,j}$ refers to the entry of row i and column j of M. Assume that we would like to know about M as precisely as possible but the only information available about M is a set of observed entries $M_{i,j}$, where $(i,j) \in \Omega$ and $\Omega \subseteq \{1,\ldots,m\} \times \{1,\ldots,n\}$. Let $\mathcal{P}_\Omega : \mathbb{R}^{m \times n} \mapsto \mathbb{R}^{m \times n}$ be the orthogonal projection onto the subspace of matrices that vanishes outside of Ω (i.e., $(i,j) \in \Omega$ *iff* $M_{i,j}$ is observed), i.e., $Y = \mathcal{P}_\Omega(X)$ is defined by $Y_{i,j} = X_{i,j}$ *if* $(i,j) \in \Omega$ and $Y_{i,j} = 0$ *otherwise*.

Therefore, the information available about M is summarized by $\mathcal{P}_\Omega(M)$. M can, in principle, be recovered from $\mathcal{P}_\Omega(M)$ by solving the nuclear norm minimization problem

$$\min ||X||_* = \sum_k \sigma_k(X) \text{ s.t. } X_{i,j} = M_{i,j} \quad \forall (i,j) \in \Omega \tag{1}$$

where, $\sigma_k(X)$ are singular values of a matrix X and can be computed via singular value decomposition (SVD) of X: $X = \sum_{k=1}^{r} \sigma_k \mathbf{u_k} \mathbf{v_k^*}$, where r is the rank of X, the $\mathbf{u_k}$'s and $\mathbf{v_k}$'s are the left and right singular vectors of X. Since nuclear norm is a convex function, it can be optimized efficiently in Equation (1) via semidefinite programming [6]. Furthermore, if Ω is sampled uniformly at random among all subset of cardinality m, and M obeys a low coherence condition, then with large probability, the unique solution to Equation (1) is exactly M. As a result, it is claimed by Candès and Recht [6]:

Remark 1. *Low-rank or approximately low-rank matrices can be recovered as the solution to Equation* (1) *using exact matrix completion via convex optimization.*

Note that it is important that Ω is sampled *uniformly* without replacement to avoid trivial situations in which a row or a column is not observed, since matrix completion is clearly impossible in such cases. Furthermore, the rank of a matrix M corresponds to the maximal number of its linearly independent rows of M.

3 PROBLEM DEFINITION

For the sake of simplicity, we adopt the concept of home device scheduling, in which we aim to schedule devices in time. Our model can be adopted to other settings in preference-dependent scheduling (PDS) problems without the loss of generality, for example, devices in home device scheduling can be seen as meetings in PDS problems. As such, our preference elicitation problem is defined formally as:

Definition 3.1. A *preference elicitation problem* is a tuple $\langle A, T, \boldsymbol{Q}, B(\cdot), \mathcal{B} \rangle$, where

- $A = \{1,2,\ldots,|A|\}$ is the set of devices and $T = \{1,2,\ldots,|T|\}$ is the set of discrete time steps, whose preferences need to be elicited;
- $\boldsymbol{Q}$ is a finite set of questions that can be asked;

Time Steps	1	2	3	4	5	6	7	8	9	10	11	12
Light #1	4	1	1	7	4	8	8	2	5	5	7	8
Kitchen Outlet	1	1	4	7	2	8	6	3	6	8	3	1
Light #2	5	2	1	6	4	9	8	1	5	5	7	8

Figure 1: A preference matrix for home device scheduling.

- $B(\cdot) : 2^{\mathcal{Q}} \rightarrow \mathbb{R}^+$ is the bother cost function that takes as input a sequence of questions and determines the bother cost of asking those questions; and
- $\mathcal{B} > 0$ is the bother cost budget (i.e., the maximum amount of bother cost that can be incurred).

Preference Matrix: The preferences of a user for having devices A scheduled in T is modeled as a matrix M, called *preference matrix*, where each row corresponds to a device $a \in A$ and each column corresponds to a time step $t \in T$[2]. We use $p_i(j) = M_{i,j}$ to denote the preference of the user having device $i \in A$ scheduled at time step $j \in T$, where the greater the value of $p_i(j)$, the more likely the user will run device i at j. For simplicity and without loss of generality, we assume that these preferences are integers in $[1, \ldots, 10]$. Figure 1 illustrates an example preference matrix M for the smart device scheduling problem with 3 devices (i.e., $|A| = 3$) and 12 time steps (i.e., $|T| = 12$). $M_{2,6} = 8$ and $M_{2,5} = 2$ mean that the user prefers to use the kitchen outlet at time step 6 than at time step 5.

As a row in M represents the scheduling preference of a device, the linear dependency between two rows a and b of M imply that the user has similar preferences on a and b. In preference-dependent scheduling (PDS) problems, including the DSM application, it is reasonable to assume that tasks (smart devices in DSM) that need to be scheduled are interdependent (see e.g., [40]). This implies that the *preference matrix is approximately low rank*, which allows us to perform exact matrix completion via convex optimization to recover the preference matrix (see Remark 1).

Preference Elicitation Questions: Each question $q \in \mathcal{Q}$ is associated with:

- a *cognitive bother cost* $c(q) > 0$ (i.e., cognitive effort required of the user to answer the question).
- a set of entries $f(q) \subseteq \{(i,j) \mid i \in A, j \in T\}$ and a set of values $\{p_i^q(j) \mid (i,j) \in f(q)\}$.

Intuitively, given a user's response to a question $q \in \mathcal{Q}$, the system fills a preference value $p_i^q(j)$ into the entry (i,j) of the preference matrix M (i.e., $M_{i,j} = p_i^q(j)$) for each element of $f(q)$. We assume that users are honest in that when two different questions fill some common entries, the preference values are consistent, i.e., $\forall q_1, q_2, (i,j) \in f(q_1) \cap f(q_2) : p_i^{q_1}(j) = p_i^{q_2}(j)$. Therefore, for convenience, we omit the superscript q in $p_i^q(j)$ from here on.

Example 3.2. We provide here two preference elicitation questions q_1 and q_2 with respect to the preference matrix in Figure 1, assuming that a day starts in time step 1 and ends in time step 12 (i.e., each time step corresponds to 2 hours).

- q_1: How likely are you to use the kitchen outlet from 12-2pm?
- q_2: How likely is that you would turn on the light #1 from 4-8pm?

where we use the following scale:

HIGHLY UNLIKELY									EXTREMELY LIKELY
1	2	3	4	5	6	7	8	9	10

Since time step 7 represents the time between 12-2pm, and time steps 9 and 10 represent the time between 4-8pm, we have $f(q_1) = \{(2,7)\}$ and $f(q_2) = \{(1,9),(1,10)\}$. In addition, a possible bother cost function is one that assigns $c(q_1) = 2$ and $c(q_2) = 3$, as a user is likely to be less bothered answering a single question that asks the preferences over two time steps than two questions that each asks the preference for a single time step. Hence, $c(q_2) \leq 2c(q_1)$.

Given a question $q \in \mathcal{Q}$ and preference matrix M, $M(q)$ is *the update of M by q* based on the user's response to q and is defined as $M(q)_{i,j} = p_i(j)$ *if* $(i,j) \in f(q)$ and $M(q)_{i,j} = M_{i,j}$ *otherwise*. For a sequence of questions $Q = \langle q_1, q_2, \ldots, q_\ell \rangle$, let $f(Q) = \bigcup_{q \in Q} f(q)$ and $M_1 = M(q_1)$, $M_2 = M_1(q_2)$, …, $M(Q) = M_\ell = M_{\ell-1}(q_\ell)$, i.e., $M(Q)$ is the result of recursively updating M by questions q_1, $q_2, \ldots, q_\ell$. It is easy to see that the order of questions by which a matrix is updated does not affect the resulting matrix.

Bother Cost Model: Our proposed framework is generic and therefore it is compatible with any bother cost model. However, in this paper, we use the model proposed by Fleming [15] since *(i)* it has been used in a large body of literature (e.g., [9, 30]), and *(ii)* it also gives the bother cost function with more exponential and logarithmic appearances for more unwilling and willing users, respectively, that fits the users' typical behavior in smart home domains [39].

Let $Q \subseteq \mathcal{Q}$ be a sequence of questions that has been asked thus far. The bother cost model by Fleming [15] defines "*bother cost so far*" (BSF) as

$$BSF_Q = \sum_{q \in Q} c(q)\beta^{e(q)} \tag{2}$$

where $0 < \beta \leq 1$ is a discount factor used to represent the diminishing impact of interactions over time (i.e., questions asked long ago will be less bothersome than questions recently asked) and $e(q)$ is the amount of elapsed time since q was asked. The total *bother cost* is then computed as

$$B(Q) = Init + \frac{1 - \alpha^{BSF_Q}}{1 - \alpha} \tag{3}$$

where[3] $\alpha = 1.26 - 0.05w$, $Init = 10 - w$, and w denotes the *willingness* of a user to interact (i.e., answering questions) on a scale of 0 (for unwilling users) to 10 (for willing users).

In our application, we assume that the system will ask questions consecutively and will thus use the number of questions that has been asked after asking q as the value for $e(q)$. For example, assume the sequence of questions $Q = \langle q_1, q_2, q_3 \rangle$, then $e(q_1) = 2$, $e(q_2) = 1$, and $e(q_3) = 0$. With $\beta = 0.95$ and $c(q_i) = 1$ for $1 \leq i \leq 3$, we have $BFS_Q = 1 + 0.95^1 + 0.95^2 = 2.8525$.

Objective Function: Let M be a preference matrix, where all entries are initially set to *null* and $M(Q)$ be the update of M by a

[2] It is worth noting that the periodicity of the device usage's preferences can also be represented by preference matrices and, thus, can be handled by our approach.

[3] Intuitively, α is intended to give a nearly linear bother curve for users with moderate willingness values (i.e, $w = 5$) while giving bother curves with more exponential and logarithmic appearances for more unwilling and willing users, respectively. The value of $Init$ is intended to reflect the cost of bothering a user for the first time in which $Init$ will be negligible (resp. quite high) for a very willing (resp. an unwilling) user.

sequence of questions $Q \subseteq \mathcal{Q}$. Our goal is to estimate the *null* entries of $M(Q)$ using the non-*null* entries in $M(Q)$ using matrix completion techniques described in the background. Given a matrix completion algorithm $\mathcal{L}$, let $\widehat{M^{\mathcal{L}}(Q)}$ be resulting estimated matrix by $\mathcal{L}$ with input $M(Q)$. When $\mathcal{L}$ is unspecified or clear from the context, we will omit it from the superscript.

Finally, the goal of the problem is to identify an optimal sequence of questions Q^* from $\mathcal{Q}$ and matrix completion algorithm $\mathcal{L}^*$:

$$\langle Q^*, \mathcal{L}^* \rangle = \arg\min_{Q,\mathcal{L}} \frac{\overbrace{||\mathcal{M} - \widehat{M^{\mathcal{L}}(Q)}||_1}^{W(Q,\mathcal{L})}}{\underbrace{|A| \times |T| - |f(Q)|}_{Z(Q)}} \text{ s.t. } B(Q) \leq \mathcal{B} \tag{4}$$

where $\mathcal{B}$ is the bother cost budget; $\mathcal{M}$ is the true (oracle) preference matrix that can be achieved in the ideal scenario where preferences for *all* devices at every time step should be elicited; and $||X||_1$ is the L_1 norm (i.e., sum of all absolute values of entries) of matrix X. In Equation (4), $W(Q, \mathcal{L})$ denotes the differences between two matrices $\mathcal{M}$ and $\widehat{M^{\mathcal{L}}(Q)}$, and $Z(Q)$ denotes the number of unfilled entries in $M(Q)$.

4 iPLEASE SYSTEM

Our approach, iPLEASE, identifies solutions of Equation (4) by using a heuristic to identify Q^*, and then employing the "matrix completion via convex optimization" (denoted with *CO* from here on) for algorithm $\mathcal{L}^*$. The reasons behind this choice are: (*i*) a matrix is completed as a solution to a nuclear norm minimization problem (see Equation (1)); (*ii*) as discussed by Candès and Recht [6] the matrix completion problem can be formulated in terms of semidefinite programming (e.g., see [41]), and there exist many efficient algorithms and high-quality softwares for solving these types of problems.

We propose two heuristics to identify Q^*:

- *One-shot* heuristic is a wholistic approach that chooses a subset of questions that provides the largest total utility given the initial partially-filled preference matrix, subject to the bother cost is no larger than the bother cost budget.
- *Multi-shot* heuristic is a greedy myopic approach that repeatedly chooses the question that provides the largest utility given the current partially-filled preference matrix, subject to the bother cost of the question is no larger than the remaining bother cost budget. In each iteration, the utility of questions will be updated.

We next describe how to compute Q^*. Assume that we have a preference matrix M that might be partially filled.

One-shot Heuristic: We formalize the problem of identifying Q^* by formulating it as a 0-1 Knapsack Problem [28], denoted with 0-1 KP. This formalization requires the utilities of the questions in $\mathcal{Q}$. Given that iPLEASE uses CO as its matrix completion algorithm, and CO assumes that the set Ω of observed entries is sampled *uniformly* without replacement, it is reasonable to assume that $f(Q)$ exhibits a uniform distribution.

Based on this assumption, the utility of a (partially-filled) matrix should be defined based on two criteria: *(1)* The larger the number of non-*null* entries in the matrix, the larger its utility; and *(2)* Populating an entry in a row or column with few non-*null* entries results in a larger increase in utility than populating an entry in a row or column with many non-*null* entries. Sigmoid functions satisfy these two criteria and are widely used to define human-decision-based utility (e.g., [12] and [21]). We thus follow the literature and also apply sigmoid functions to estimate the utility of a (partially-filled) matrix, and the utility of a question is defined as the difference in the utility of the preference matrices before and after asking that question.

Definition 4.1 (Utility of a matrix). Let M be a (partially-filled) matrix of size $m \times n$. The utility of M, denoted with U_M, can be estimated as a sigmoid function

$$U_M = \sum_{1 \leq i \leq m} \frac{n}{1 + e^{-(rfill_{M,i} - 1)}} + \sum_{1 \leq j \leq n} \frac{m}{1 + e^{-(cfill_{M,j} - 1)}} \tag{5}$$

where $rfill_{M,i}$ (resp. $cfill_{M,j}$) is the number of non-*null* entries in the row i (resp. column j) of M.

The utility of a question q is defined along with a matrix (i.e., the current matrix before asking q).

Definition 4.2 (Utility of a question). Let M be a matrix. The utility of a question q, denoted with $U_M(q)$, is defined as: $U_M(q) = U_{M(q)} - U_M$.

Because the sigmoid function is *monotonic* and updating M by q will not increase the number of *null* entries, $U_M(q) \geq 0$. It is easy to see that the following holds.

Proposition 1. *For $q_1 \neq q_2$, $U_{M(\{q_1,q_2\})} \leq U_{M(q_1)} + U_{M(q_2)}$. The equality happens when $f(q_1) \cap f(q_2) = \emptyset$*

Since the constraint in 0-1 KP is linear, we need to estimate the bother cost of a sequence of questions as a linear equation. $BSF_Q^{\beta=y}$ be the BSF_Q in which $\beta = y$. From Equation (2), since $0 < \beta \leq 1$, it is straightforward to have the following lemma.

Lemma 4.3. *Let $Q \subseteq \mathcal{Q}$ be a sequence of questions that has been asked, and given a BSF_Q with respect to an arbitrary $\beta (0 < \beta \leq 1)$. Then, $BSF_Q \leq BSF_Q^{\beta=1}$.*

From Equation (3), it is straightforward to show:

Lemma 4.4. *Assume a bother cost budget $\mathcal{B} > 0$, and let $Q \subseteq \mathcal{Q}$ be a sequence of questions that has been asked. Then, $B(Q) \leq \mathcal{B}$ iff*

$$BSF_Q \leq \frac{log_{10}\Big(1 - (\mathcal{B} - 10 + w)(1 - \alpha)\Big)}{log_{10}\alpha} \tag{6}$$

Intuitively, Lemma 4.4 says that the "bother cost so far" of Q satisfies Equation (6) if and only if the total bother cost $B(Q)$ does not exceed the bother cost budget $\mathcal{B}$.

Let $T_{\mathcal{B},w}$ denote the right-hand side of Equation (6). Based on Lemmas 4.3-4.4, we have the following theorem.

Theorem 4.5. *Given a bother cost budget $\mathcal{B} > 0$ and a sequence of questions $Q \subseteq \mathcal{Q}$ that has been asked, if $BSF_Q^{\beta=1} \leq T_{\mathcal{B},w}$, then $B(Q) \leq \mathcal{B}$.*

Since $T_{\mathcal{B},w}$ can be seen as a constant which depends on the inputs $\mathcal{B}$ and w (since α is computed from w), Theorem 4.5 allows us to check whether $B(Q) \leq \mathcal{B}$ by checking $BSF_Q^{\beta=1} \leq T_{\mathcal{B},w}$,

Algorithm 1: one-shot($\mathcal{Q}, M, \mathcal{B}, w$)

Input : A set of questions $\mathcal{Q}$; a matrix M; a bother cost budget $\mathcal{B}$; a willingness value w
Output : A set of question $Q^* \subseteq \mathcal{Q}$
1 **return** Q^* = 0-1KP($\mathcal{Q}, M, \mathcal{B}, w$)

Algorithm 2: multi-shot($\mathcal{Q}, M, \mathcal{B}, w$)

Input : A set of questions $\mathcal{Q}$; a matrix M; a bother cost budget $\mathcal{B}$; a willingness value w
Output : A set of question $Q^* \subseteq \mathcal{Q}$
2 Let $Q^* = \{\}$
3 **while** *true* **do**
4 Compute $\mathcal{B}' = remain(Q, \mathcal{B}, \beta, w)$
5 Compute q = one-shot-one-question($\mathcal{Q}, M, \mathcal{B}', w$)
6 **if** $q \neq null$ **then** $Q^* = Q^* \cup \{q\}$
7 **else return** Q^*

making it possible to identify Q^* using the 0-1 KP. This is done as follows.

Given the set of questions $\mathcal{Q} = \{q_1, \ldots, q_n\}$, a matrix M, a bother cost budget $\mathcal{B}$, and a willingness value w, the 0-1 Knapsack Problem for selecting a subset of questions $Q \subseteq \mathcal{Q}$ such that

$$\text{maximize} \quad \sum_{k=1}^{n} U_M(q_k) x_k \tag{7}$$

$$\text{subject to} \quad \sum_{k=1}^{n} c(q_k) x_k \leq T_{\mathcal{B},w} \tag{8}$$

where, for $1 \leq k \leq n$

$$x_k = \begin{cases} 1, & \text{if question } q_k \in Q \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

We write Q = 0-1KP($\mathcal{Q}, M, \mathcal{B}, w$) to indicate that Q is a solution of the above problem, and $Q = \emptyset$ when the problem has no solution.

Algorithm 1 shows the pseudo-code of using the one-shot heuristic given a set of questions $\mathcal{Q}$, a matrix M, a bother cost budget $\mathcal{B}$, and a willingness value w. It selects a subset of question Q^*, denoted with Q^* = one-shot($\mathcal{Q}, M, \mathcal{B}, \beta, w$) based on this heuristic and returns that subset as the solution of the preference elicitation problem.

Note that $\sum_{k=1}^{n} c(q_k) x_k$ in Equation (8) is identical to $BSF_{Q^*}^{\beta=1}$ where Q^* = 0-1KP($\mathcal{Q}, M, \mathcal{B}, w$). Theorem 4.5, the knapsack constraint (8), and Algorithm 1 imply the next theorem.

Theorem 4.6. *If* Q^* = *one-shot*($\mathcal{Q}, M, \mathcal{B}, w$) *then* $B(Q^*) \leq \mathcal{B}$

Multi-shot Heuristic: The one-shot heuristic is simple but it has some undesirable consequences. It chooses a subset of questions based on utilities of questions that depend solely on the current (partially-filled) matrix. Therefore, it does not take into account of the mutual effect of questions in that subset. In particular, in an extreme case, if questions in that subset elicit too many cells of the same rows or of the same columns, by Proposition 1, the objective function in Equation (7) will overly estimate the actual improved utility of preference matrix after asking that subset of questions.

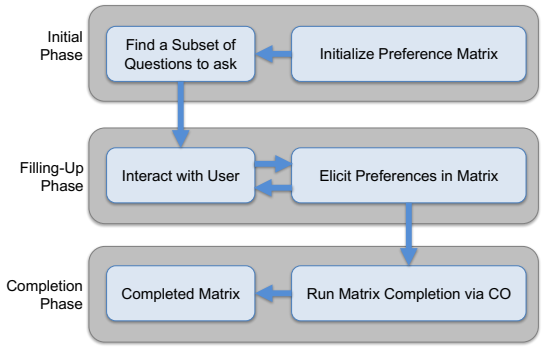


Figure 2: iPLEASE Workflow

This motivates the *multi-shot* heuristic that repeatedly chooses one question at a time until the budget cost is exhausted. In this paper, we will use a greedy algorithm for implementing this heuristic, i.e., at any step, we select the question that provides the largest updated utility given the current partially-filled preference matrix, subject to the constraint that the bother cost of that question is no larger than the remaining bother cost budget.

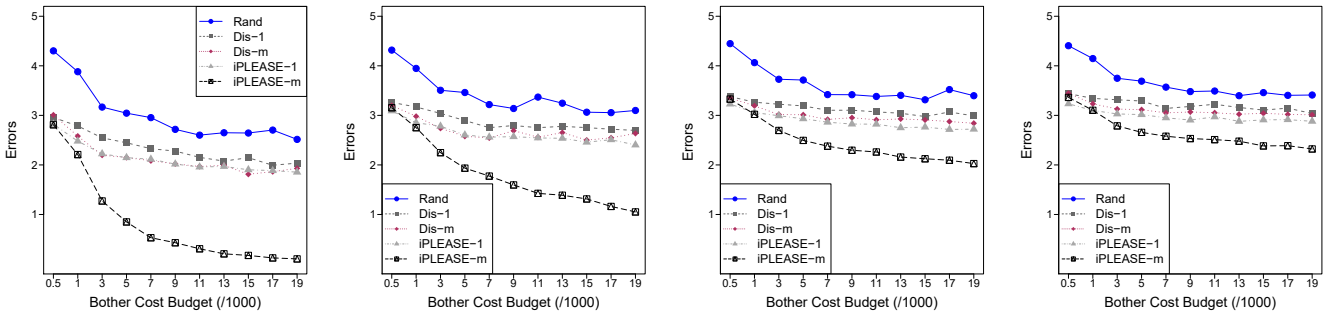
Given $\mathcal{Q}$, M, $\mathcal{B}$, β, and w, as in the one-shot heuristic formalization, we define the *one-shot-one-question* problem as the problem of selecting one question $q \in \mathcal{Q}$ such that $U_M(q) = \max\{U_M(q) \mid q \in \mathcal{Q}, c(q) \leq \mathcal{B}\}$. When no such q exists, we write $q = null$. We also write q = one-shot-one-question($\mathcal{Q}, M, \mathcal{B}, w$) to denote that q is a solution of this problem.

Let Q be a sequence of questions $\langle q_1, \ldots, q_k \rangle$. We denote with $remain(Q, \mathcal{B}, \beta, w)$ the remaining bother cost budget after asking the questions in Q, i.e.,

$$remain(Q, \mathcal{B}, \beta, w) = T_{\mathcal{B},w} - \sum_{q_k \in Q} c(q_k) \beta^{e(q_k)+1} \tag{10}$$

Algorithm 2 shows the pseudo-code of using the multi-shot heuristic given a set of questions $\mathcal{Q}$, a matrix M, a bother cost budget $\mathcal{B}$, and a willingness value w. It selects a subset of question Q^*, denoted with Q^* = multi-shot($\mathcal{Q}, M, \mathcal{B}, \beta, w$) based on this heuristic and returns that subset as the solution of the preference elicitation problem. In more detail, it first initializes Q^* as an empty set (Line 2), and then iteratively adds to Q^* a question q that is the solution of the problem one-shot-one-question($\mathcal{Q}, M, \mathcal{B}', w$) in which $\mathcal{B}'$ is updated as $\mathcal{B}' = remain(Q, \mathcal{B}, \beta, w)$ (Lines 3-7). This iterative update ends when one-shot-one-question($\mathcal{Q}, M, \mathcal{B}', w$) returns *null* (i.e., all questions that are not in Q^* have a bother cost that is larger than the remaining bother cost budget). It is straightforward to see that this algorithm is guaranteed to terminate since the bother cost of all questions is positive and the remaining bother cost budget is reduced after each iteration.

iPLEASE Workflow: Given $\langle A, T, \mathcal{Q}, B(\cdot), \mathcal{B} \rangle$, a diminishing factor β, and a willingness value w of a user to interact with the bother cost function defined in the previous section, our iPLEASE system has an extra parameter, an integer $\mathcal{I}$. It uses this parameter to *initialize* the preference matrix M with a *random* set $I \subseteq \{1, \ldots, |A|\} \times \{1, \ldots, |T|\}$ of positions in M such that $|I| = \mathcal{I}$ before focusing on selecting the set of questions that will be used to complete M. This is a reasonable



(a) Rank = 5 (b) Rank = 10 (c) Rank = 20 (d) Rank = 30

Figure 3: Error Results on Randomly Generated Problems

assumption as it is acceptable for a system to ask users to answer questions in a survey prior to the first time in using the system. We consider this initialization process (see Initial Phase below) as a preprocessing phase to pre-populate the preference matrix. The workflow of iPLEASE (see Figure 2) has the following phases:

- *Initial Phase*: iPLEASE solves the 0-1 integer linear programming problem defined by: Select a subset of questions Q that fills positions in I in M such that

$$\text{minimize} \sum_{k=1}^{n} c(q_k)x_k \tag{11}$$

$$\text{subject to} \sum_{k=1}^{n} c(q_k)x_k \leq T_{\mathcal{B},w} \tag{12}$$

$$\sum_{k=1}^{n} y_{i,j}x_k \geq 1 \quad \forall (i,j) \in I \tag{13}$$

  where, for $1 \leq k \leq n$, x_k is defined in Equation (9), and

$$y_{i,j} = \begin{cases} 1, & \text{if } q_k \text{ is selected} \wedge q_k \in g(i,j) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

  where $g(i,j) = \{q \in \mathcal{Q} \mid (i,j) \in f(q)\}$.
  We write $Q = Initial\_Fill(\mathcal{Q}, I, \mathcal{B}, w)$ to say that Q is the solution of the initial phase. When there is no solution, $Q = \{\emptyset\}$.
- *Filling-Up Phase*: Assume that $Q = Initial\_Fill(\mathcal{Q}, I, \mathcal{B}, w)$. If $Q \neq \{\emptyset\}$, iPLEASE uses the one-shot or multi-shot heuristic to fill the matrix M under the condition that the total budget cost is $\mathcal{B}' = \mathcal{B} - B(Q)$.
- *Completion Phase*: iPLEASE uses CO to complete M, the result of the second phase whose result is $\widehat{M}$.

5 EXPERIMENTAL RESULTS

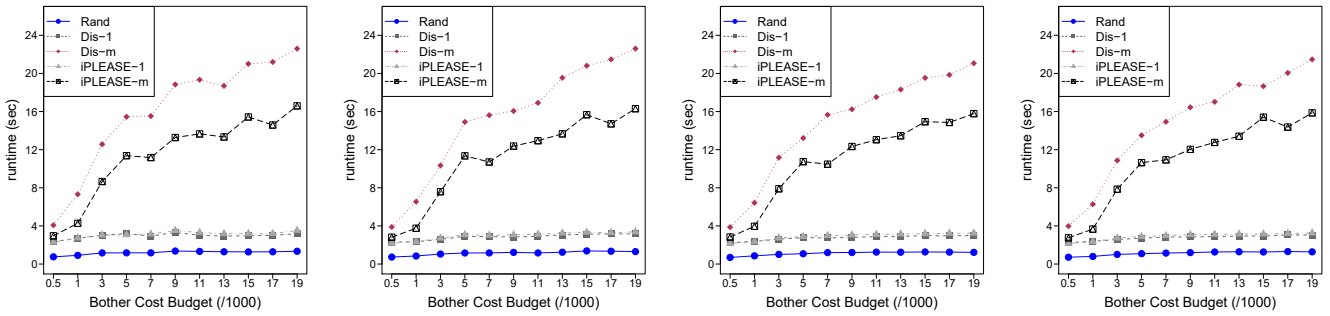
We empirically evaluate the iPLEASE framework on randomly-generated problems (i.e., randomly-generated preference low rank matrices) as well as demand-side management (DSM) problems using the real-world REDD dataset [22].

We implemented iPLEASE with the one-shot heuristic (iP-1) and iPLEASE with the multi-shot heuristic (iP-m) using MATLAB® Release 2016b, in which we used CVX, a package for specifying and solving convex programs [11, 18], to solve the nuclear norm minimization problem in Equation (1). To the best of our knowledge, there is no algorithm that directly solves our problem described in Equation (4). Therefore, we implement three benchmark elicitation frameworks to compare against our proposed framework:

- **Random (Rand)**: This framework iteratively asks the user a *random* (non-duplicated) question from the given set of questions until the remaining bother cost budget is insufficient to ask any other question. Then, it uses CO, the same matrix completion algorithm used by iPLEASE, to complete the resulting partially-filled matrix.
- **One-shot Disagreement (Dis-1) and Multi-shot Disagreement (Dis-m)**: The Dis-1 and Dis-m frameworks consist of 3 phases that are the same as the 3 phases in iP-1 and iP-m, respectively. The only exception is that they compute the utility of questions differently, based on the level of *disagreement* proposed by Chakraborty et al. [8] and Lan et al. [24]. Specifically, a committee of matrix completion algorithms – that consists of CO [6] and "Matrix completion from a few entries" [20][4] – are applied on the partially-filled matrix from Step 2 to impute *null* entries. The *variance* of imputing (among committee members) of each entry is taken as a measure of uncertainty of that entry. The utility of a question is computed as in Definition 4.2 in which the utility of a matrix is the summation of the uncertainty of all of its entries, which is different from Definition 4.1.

In our experiments, we set $|A| = 55$, $|T| = 24$, $\beta = 1$, $w = 5$, and $\mathcal{I} = 150$ (see Further Discussion subsection later for the choice of $\mathcal{I}$). Then, the size of the preference matrix is 55×24. We created 1790 questions (i.e., $|\mathcal{Q}| = 1790$). Intuitively, each question straightforwardly asks the user their preference for using k devices in ℓ time steps (i.e., $1 \leq k \leq 55, 1 \leq \ell \leq 24$). The cost of each question q is set to the following: $c(q) = k$ if $\ell = 1$ and $c(q) = k \times \lfloor \ell/2 \rfloor$ otherwise. Its intuition is that each different device will require 1 unit of cognitive cost to answer. The component $\lfloor \ell/2 \rfloor$ intuitively reflects our assumption that the user is less bothered answering one question that asks for preferences on multiple time steps compared to answering multiple questions, each of which asks for the preference for one time step. In our experimental setup, the range of cognitive costs of the 1790 questions is $[1, 72]$. We report the error in the solutions found as well as the runtimes of the different algorithms, averaged over 30 randomly generated instances per configuration. The error in a solution is computed using the

[4]We use a publicly-available implementation found at goo.gl/9EnrXN



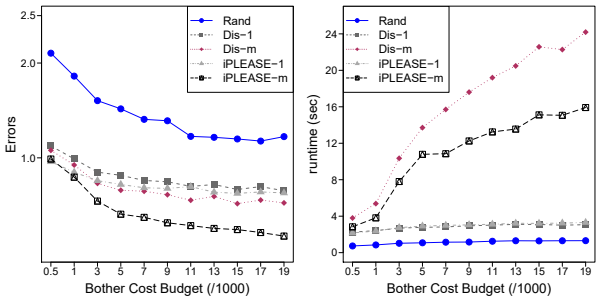
(a) Rank = 5 (b) Rank = 10 (c) Rank = 20 (d) Rank = 30

Figure 4: Runtimes on Randomly Generated Problems

objective function in Equation (4), i.e., $||W(Q,\mathcal{L})||_1/Z(Q)$. All experiments are performed on 2.8 GHz Intel Core i7 machine with 16GB of memory.

Randomly Generated Problems: We vary the rank of the oracle preference matrix, denoted with *rank* and the bother cost budget. The r-rank oracle preference matrix is generated as follows. First we generate a matrix $\mathcal{M}^0$ of size $r \times 24$ whose entries' value are randomly generated from the range $[1, 10]$, and thus, rank of $\mathcal{M}^0$ is r. Then the oracle preference matrix of size 55×24 is generated in the way that each of its row is selected randomly from rows of $\mathcal{M}^0$. In all experiments on random dataset, we make sure that the oracle preference matrices have the rank as indicated. Figure 3 and Figure 4 show the average error and runtime, respectively. We make the following observations:

- As expected, for all algorithms, the solutions improve (their error decreases) as the budget $\mathcal{B}$ increases. The reason is that the algorithms are able to ask more questions, which results in the need to approximate fewer entries in the preference matrix.
- In general, the ordering of algorithms from best to worst (in terms of the quality of solutions found) is iP-m, iP-1, Dis-m, Dis-1, and Rand. iP-m is consistently better than the others, and Rand is consistently worse than the others. iP-1, Dis-m, and Dis-1 algorithms find similar solutions with no statistically significant differences in quality. This observation shows that the use of sigmoid functions to define the utility of questions (see Definitions 4.1 and 4.2) coupled with the ability of iP-m to take into account the mutual effect of questions by updating the utility of questions in each iteration results in statistically improved results.
- The error for iP-m error increases with increasing rank, and the error for the other algorithms remain relatively unchanged for all ranks.
- As expected, for all algorithms, the runtimes increase as the budget $\mathcal{B}$ increases. The reason is that the algorithms need to identify more questions to ask as the budget increases.
- The increase in runtime is negligible for Rand, iP-1, and Dis-1. The reason is that the additional computation needed to identify the additional questions is minimal.
- The increase in runtime is significant for iP-m and Dis-m as they need to re-evaluate the utility of all unchosen questions in each iteration in addition to choosing an additional question to ask.



(a) Error Results (b) Runtime Results

Figure 5: Experiment Results on REDD Dataset

- In general, the ordering of algorithms from fastest to slowest is Rand, iP-1, Dis-1, iP-m, and Dis-m. The runtimes of iP-1 and Dis-1 are statistically similar. Rand is the fastest as it does not need to compute the utility of questions. The one-shot algorithms, iP-1 and Dis-1, are both faster than their multi-shot counterparts as they need to compute the utility of questions once only. In contrast, the multi-shot algorithms, iP-m and Dis-m, both need to update the utility of questions repeatedly in each iteration. Finally, iP-m is faster than Dis-m because it needs to run the matrix completion algorithm once only. In contrast, Dis-m needs to run all the matrix completion algorithms in its committee of algorithms in each iteration.

Therefore, in summary, iP-m is able to find the best solution but at a cost of large runtimes. In contrast, iP-1 and Dis-1 are able to find worse solutions at smaller runtimes. This range of algorithms thus allow users to trade off solution quality for smaller runtimes based on the requirements in their applications.

REDD Dataset: We use the Reference Energy Disaggregation Data Set (REDD) [22], which includes electrical usage data from six different houses for approximately 35 days. We use data of 9 devices in house 1, collected from April 18, 2011 to May 24, 2011, as they provide the most detailed data. The raw data in the dataset contains power consumption of devices with a granularity of 3 seconds, which we converted to a list of cyclic on-off events. If a device i is turned on or off at a time step j on a specific date d, the preference for using i at time step j on date d is set to 8 or 2, respectively.

We then average the preferences over all dates. We first create a matrix $\mathcal{M}^1$ of size 9×24 where each row is the preferences for using 1 (out of 9) device for 24 time steps. We then generate the oracle preference matrix of size 55×24, which has rank of 9, in the way that each of its rows is selected randomly from rows of $\mathcal{M}^1$. Figure 5 shows the average error and runtime, where, in general, the trends from the randomly generated problems apply here as well. However, the magnitude of the errors in this dataset is up to 50% smaller than in the randomly generated problems. The reason is likely due to the underlying strong interdependencies between the preferences – the *approximate* rank of the dataset is around 5.

Further Discussions: We discuss here the intuition of setting $\mathcal{I} = 150$ in our experiments. We observe that if $\mathcal{I}$ is set to a smaller value, e.g., $\mathcal{I} = 100$, "Matrix completion from a few entries" in Dis-1 and Dis-m takes too long to converge and fails to converge at times. Moreover, if we set $\mathcal{I}$ to a larger value, for experiments with $\mathcal{B} = 500$, the remaining bother cost budget for computing heuristics in Filling-Up Phase of iPLEASE is very small and, thus, it will not clearly show the advantage of our proposed heuristics.

6 RELATED WORK

As iPLEASE performs active matrix completion to solve preference elicitation problems, we describe how it relates to these two broad areas. There is a large body of research on *active matrix completion*: Boutilier et al. [5] used collaborative filtering to query ratings for products to maximize the increase in the *expected value of information*. Extending this work, Jin and Si [19] assumes prior information, and proposed a Bayesian approach that query entries to minimize model uncertainty. In a different approach, Rish and Tesauro [31] suggested a margin-based approach that queries entries with the *least completion confidence* that are closest to the decision boundary of the completion model. Chakraborty et al. [8] proposed distribution- and committee-based querying strategies, which were later extended by Sutherland et al. [34] to strategies that minimize the uncertainty of the completed model, using probabilistic matrix factorization methods. However, unlike iPLEASE, they all do not consider bother costs in their querying strategies.

There is also a large body of research on *preference elicitation* [17]. Due to space restrictions, we focus on the techniques that are most closely related to our approach. They include *passive* elicitation techniques—that make use of machine learning methods to learn users' preferences based on historical data [27, 39]—and static elicitation techniques that either asks users a number of preset questions [38] as well as alerts and notification messages to interact with users [10], or asks users to rank alternative options or user-provided option improvements to learn a (possibly approximately) user preference function [4, 7, 37, 42].

In addition, there are some lines of work that address preference-dependent scheduling problems where some preferences are missing as incomplete soft constraint problems. For example, open CSPs [13] and interactive CSPs [23] work with domains that can be partially specified. Their approaches are to solve larger and larger problems until a solution is found, while minimizing the number of variable values asked to the user. Differently, Gelain et al. [16] assumes variable values are known from the beginning, while some preferences may be missing, and their general solver schema is to interleave branch and bound search with elicitation steps.

However, all the works above neither consider heterogenous bother costs nor interdependency between the preferences. Thus, they cannot efficiently tackle our problem. There are two recent frameworks that do consider bother costs: Tabakhi et al. [35] uses heuristics to identify k critical devices whose preferences should be elicited, where k is a user-defined parameter. Thus, it assumes that all queries have identical bother costs, which is not realistic. Truong et al. [39] do consider heterogenous bother costs, but do not assume any underlying interdependency structure of the preferences and, thus, cannot exploit it. Finally, it is worth to mention a very recent related work of Lewenberg et al. [25], which uses Bayesian matrix factorization to infer interdependent data in a surveying domain. However, this work does not consider user bother cost during the inference process and, thus, is not suitable to our problem.

Additionally, in the automated negotiation literature, there exist some work in which queries to ask users can be associated with arbitrary bother costs [1, 2]. However, bother costs are typically included in the objective function. Thus, proposed solutions typically select a query that ensures the highest expected negotiation payoff (i.e., a combination of utility and bother cost). As such, it is regarded as a one-shot optimization problem. In contrast, iPLEASE selects the question to ask based on the current state of partially filled preference matrix and the remaining predefined-bother cost budget. Thus, we are more interested in an optimal sequence of queries whose total bother cost does not exceed a budget.

7 CONCLUSIONS

In preference-dependent scheduling (PDS) problems, user preferences need to be elicited or approximated prior to solving the scheduling problem. As PDS systems cannot have an unlimited amount of interactions with users, preference elicitation algorithms seek to identify the best subset of questions to ask users such that the most useful information is gained. Existing methods assume that all questions have equal bother costs, which is unrealistic in practice. Further, they do not exploit the fact of often having strong inter-dependencies between preferences for tasks that need to be scheduled.

In this paper, we introduce the iPLEASE system, which remedies both of these deficiencies by (1) incorporating a bother cost to each question, which is dependent on the amount of information it elicits, and (2) using matrix completion algorithms, which exploits the inter-dependencies between tasks, to approximate the preference matrix. Experimental results show that it outperforms non-trivial benchmarks on both randomly-generated problems as well as on a real-world DSM dataset. Future work includes the integration of this framework with scheduling algorithms to better evaluate the impact of the schedules found when different preference elicitation algorithms are used.

8 ACKNOWLEDGEMENTS

This research is partially supported by NSF grants 1345232 and 1619273. In addition, Long Tran-Thanh was supported by the EPSRC funded project STRICT (EP/N02026X/1).

REFERENCES

[1] Tim Baarslag, Alper T. Alan, Richard C. Gomer, Ilaria Liccardi, Helia Marreiros, Enrico H. Gerding, and M. C. Schraefel. 2016. Negotiation as an Interaction Mechanism for Deciding App Permissions. In *Proc. of CHI, Extended Abstracts*. 2012–2019.

[2] Tim Baarslag and Michael Kaisers. 2017. The Value of Information in Automated Negotiation: A Decision Model for Eliciting User Preferences. In *Proc. of AAMAS*. 391–400.

[3] Jacek Blazewicz, Wolfgang Domschke, and Erwin Pesch. 1996. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research* 93, 1 (1996), 1 – 33.

[4] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artif. Intell.* 170, 8-9 (2006), 686–713.

[5] Craig Boutilier, Richard S. Zemel, and Benjamin M. Marlin. 2003. Active Collaborative Filtering. In *Proc. of UAI*. 98–106.

[6] Emmanuel J. Candès and Benjamin Recht. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics* 9, 6 (2009), 717–772.

[7] Urszula Chajewska, Daphne Koller, and Ronald Parr. 2000. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proc. of AAAI*. 363–369.

[8] Shayok Chakraborty, Jiayu Zhou, Vineeth Nallure Balasubramanian, Sethuraman Panchanathan, Ian Davidson, and Jieping Ye. 2013. Active Matrix Completion. In *Proc. of ICDM*. 81–90.

[9] Robin Cohen, Michael Y. K. Cheng, and Michael W. Fleming. 2005. Why bother about bother: Is it worth it to ask the user. In *Proc. of AAAI Fall Symposium*.

[10] Enrico Costanza, Joel E. Fischer, James A. Colley, Tom Rodden, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 2014. Doing the laundry with agents: a field trial of a future smart energy system in the home. In *Proc. of CHI*. 813–822.

[11] Inc. CVX Research. 2012. CVX: Matlab Software for Disciplined Convex Programming, version 2.0. http://cvxr.com/cvx. (Aug. 2012).

[12] S. Dahi and S. Tabbane. 2013. Sigmoid utility function formulation for handoff reducing Access model in cognitive radio. In *Proc. of ISCIT*. 166–170.

[13] Boi Faltings and Santiago Macho-Gonzalez. 2005. Open constraint programming. *Artif. Intell.* 161, 1-2 (2005), 181–208.

[14] Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. 2017. A Multiagent System Approach to Scheduling Devices in Smart Homes. In *Proc. of AAMAS*. 981–989.

[15] Michael William Fleming. 2004. *Reasoning About Interaction in Mixed-initiative Artificial Intelligence Systems*. Ph.D. Dissertation. Waterloo, Ont., Canada, Canada. Advisor(s) Cohen, Robin. AAINQ91996.

[16] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. 2010. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artif. Intell.* 174, 3-4 (2010), 270–294.

[17] Judy Goldsmith and Ulrich Junker. 2008. Preference Handling for Artificial Intelligence. *AI Magazine* 29, 4 (2008), 9–12.

[18] M. Grant and S. Boyd. 2008. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, V. Blondel, S. Boyd, and H. Kimura (Eds.). Springer-Verlag Limited, 95–110. http://stanford.edu/~boyd/graph_dcp.html.

[19] Rong Jin and Luo Si. 2004. A Bayesian Approach toward Active Learning for Collaborative Filtering. In *Proc. of UAI*. 278–285.

[20] Raghunandan H. Keshavan, Sewoong Oh, and Andrea Montanari. 2009. Matrix completion from a few entries. In *Proc. of ISIT*. 324–328.

[21] Miriam C. Klein-FlÂijgge, Steven W. Kennerley, Ana C. Saraiva, Will D. Penny, and Sven Bestmann. 2015. Behavioral Modeling of Human Choices Reveals Dissociable Effects of Physical Effort and Temporal Delay on Reward Devaluation. *PLOS Computational Biology* 11, 3 (03 2015), 1–31.

[22] J. Zico Kolter and Matthew J. Johnson. 2011. REDD: A Public Data Set for Energy Disaggregation Research. In *Proc. of SustKDD*.

[23] Evelina Lamma, Paola Mello, Michela Milano, Rita Cucchiara, Marco Gavanelli, and Massimo Piccardi. 1999. Constraint Propagation and Value Acquisition: Why we should do it Interactively. In *Proc. of IJCAI*. 468–477.

[24] Chao Lan, Yujie Deng, and Jun Huan. 2016. A disagreement-based active matrix completion approach with provable guarantee. In *Proc. of IJCNN*. 4082–4088.

[25] Yoad Lewenberg, Yoram Bachrach, Ulrich Paquet, and Jeffrey S Rosenschein. 2017. Knowing What to Ask: A Bayesian Active Learning Approach to the Surveying Problem. In *Proc. of AAAI*. 1396–1402.

[26] Rajiv T. Maheswaran, Milind Tambe, Emma Bowring, Jonathan P. Pearce, and Pradeep Varakantham. 2004. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling. In *Proc. of AAMAS*. 310–317.

[27] Oliver Parson, Siddhartha Ghosh, Mark J. Weal, and Alex Rogers. 2012. Non-Intrusive Load Monitoring Using Prior Models of General Appliance Types. In *Proc. of AAAI*.

[28] David Pisinger. 1997. A Minimal Algorithm for the 0-1 Knapsack Problem. *Operations Research* 45, 5 (1997), 758–767.

[29] Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nicholas R. Jennings. 2011. Agent-based homeostatic control for green energy in the smart grid. *ACM TIST* 2, 4 (2011), 35:1–35:28.

[30] Yonglin Ren, Mian Qin, and Weilin Ren. 2007. A Web Intelligent System based on Measuring the Effects of Bother. In *Proc. of WIC*. 715–718.

[31] Irina Rish and Gerald Tesauro. 2008. Active Collaborative Prediction with Maximum Margin Matrix Factorization. In *Proc. of ISAIM*.

[32] Pierre Rust, Gauthier Picard, and Fano Ramparany. 2016. Using Message-Passing DCOP Algorithms to Solve Energy-Efficient Smart Environment Configuration Problems. In *Proc. of IJCAI*. 468–474.

[33] Rahul Singh. 2003. *RCal: An Autonomous Agent for Intelligent Distributed Meeting Scheduling*. Master's thesis. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[34] Dougal J. Sutherland, Barnabás Póczos, and Jeff G. Schneider. 2013. Active learning and search on low-rank matrices. In *Proc. of KDD*. 212–220.

[35] Atena M. Tabakhi, Tiep Le, Ferdinando Fioretto, and William Yeoh. 2017. Preference Elicitation for DCOPs. In *Proc. of CP*. 278–296.

[36] Milind Tambe. 2008. Electric Elves: What Went Wrong and Why. *AI Magazine* 29, 2 (2008), 23–27.

[37] Stefano Teso, Paolo Dragone, and Andrea Passerini. 2017. Coactive Critiquing: Elicitation of Preferences and Features. In *Proc. of AAAI*. 2639–2645.

[38] Walid Trabelsi, Kenneth N. Brown, and Barry O'Sullivan. 2015. Preference Elicitation and Reasoning While Smart Shifting of Home Appliances. *Energy Procedia* 83 (2015), 389–398.

[39] Ngoc Cuong Truong, Tim Baarslag, Sarvapali D. Ramchurn, and Long Tran-Thanh. 2016. Interactive Scheduling of Appliance Usage in the Home. In *Proc. of IJCAI*. 869–877.

[40] Ngoc Cuong Truong, James McInerney, Long Tran-Thanh, Enrico Costanza, and Sarvapali D. Ramchurn. 2013. Forecasting Multi-Appliance Usage for Smart Home Energy Management. In *Proc. of IJCAI*. 2908–2914.

[41] Lieven Vandenberghe and Stephen P. Boyd. 1996. Semidefinite Programming. *SIAM Rev.* 38, 1 (1996), 49–95.

[42] Paolo Viappiani and Craig Boutilier. 2010. Optimal Bayesian Recommendation Sets and Myopically Optimal Choice Query Sets. In *Proc. of NIPS*. 2352–2360.

[43] Perukrishnen Vytelingum, Thomas Voice, Sarvapali D. Ramchurn, Alex Rogers, and Nicholas R. Jennings. 2011. Theoretical and Practical Foundations of Large-Scale Agent-Based Micro-Storage in the Smart Grid. *J. Artif. Intell. Res.* 42 (2011), 765–813.