

HITSNDIFFS: From Truth Discovery to Ability Discovery by Recovering Matrices with the Consecutive Ones Property

Zixuan Chen

Northeastern University

chen.zixu@northeastern.edu

Subhodeep Mitra

Google

mitradeep1@gmail.com

R. Ravi

Carnegie Mellon University

ravi@andrew.cmu.edu

Wolfgang Gatterbauer

Northeastern University

w.gatterbauer@northeastern.edu

Abstract—We analyze a general problem in a crowd-sourced setting where one user asks a question (also called item) and other users return answers (also called labels) for this question. Different from existing crowd sourcing work which focuses on finding the most appropriate label for the question (the “truth”), our problem is to determine a ranking of the users based on their ability to answer questions. We call this problem “ability discovery” to emphasize the connection to and duality with the more well-studied problem of “truth discovery”.

To model items and their labels in a principled way, we draw upon Item Response Theory (IRT) which is the widely accepted theory behind standardized tests such as SAT and GRE. We start from an idealized setting where the relative performance of users is consistent across items and better users choose better fitting labels for each item. We posit that a principled algorithmic solution to our more general problem should solve this ideal setting correctly and observe that the response matrices in this setting obey the *Consecutive Ones Property* (C1P). While C1P is well understood algorithmically with various discrete algorithms, we devise a novel variant of the HITS algorithm which we call “HITSNDIFFS” (or HND), and prove that it can recover the ideal C1P-permutation in case it exists. Unlike fast combinatorial algorithms for finding the consecutive ones permutation (if it exists), HND also returns an ordering when such a permutation does not exist. Thus it provides a *principled heuristic for our problem that is guaranteed to return the correct answer in the ideal setting*. Our experiments show that HND produces user rankings with robustly high accuracy compared to state-of-the-art truth discovery methods. We also show that our novel variant of HITS scales better in the number of users than ABH, the only prior spectral C1P reconstruction algorithm.

Index Terms—truth discovery, item response theory, consecutive ones property

I. INTRODUCTION

Motivation. We first present a couple of examples from a class to a more general crowdsourcing context.

Example 1 (Student ranking). *Kiyana, an innovative instructor for an online class who suffered from the leakage of previous exam questions and difficulty of creating new ones, notices a lot of interactions in student forums like Piazza [2] and pilots a new learning approach. Since students are willing to ask and answer questions, Kiyana aims to utilize such communication for both practice and assessment of the class by requiring students to suggest and answer multiple-choice questions (MCQs) themselves. The task of students*

is to come up with meaningful MCQs (including question stems and choices) related to the topics of the class and answer the questions from others. In this way, the initiative of and interactions between students are encouraged, and their performances can be used as another important measure (e.g., a “participation” score) towards the grade assessment, in addition to traditional exam scores. To assess students, Kiyana first simply counts how many times a student answers questions, which is the traditional way for an instructor to give a participation score for students in a forum. However, in this way, the grades are biased towards students who answer a lot of questions randomly. The second attempt is to require all students to answer the same number of questions and rank them by how many questions they answer correctly, which still requires a lot of efforts for her to figure out all the correct answers and suffers from the problem that each question has quite different difficulties. Kiyana wonders whether there is a more principled way for ranking students by their abilities to answer questions correctly.

Example 2 (Crowd workers ranking). *Daiyu wants to release a human intelligence task which consists of a set of questions at a crowdsourcing platform like Amazon Mechanical Turk [1]. Suffering from low-quality answers from the crowd workers, she wonders whether there is a better way to select top crowd workers instead of simply setting thresholds for the number of tasks they have finished or finished successfully.*

The above examples motivate our problem of “ability discovery” which ranks the users (students/workers) based on their abilities to answer questions correctly.

The ability discovery problem. We have m users and n items.¹ Each item has up to k labels,² and the labels usually vary between items (the items are thus “heterogeneous”). Each user chooses up to one label to each of the items, and two users may choose the same label to the same item. Our goal is to derive a principled way for determining a ranking of the users in terms of their ability to pick correct labels for the items based solely on the user responses.

¹We use item/question and label option/answer/choice interchangeably.

²In other words, the item(s) with the most unique labels has k different labels. Labels can be proposed either from questioners or answerers.

Connection to truth discovery. Ability discovery can be considered a dual problem of the widely studied *truth discovery* problem [79]. The setup is similar; the difference is that the truth discovery problem measures success in finding the truth (thus the correct label for each item) whereas our problem focuses on finding the correct ranking of the users by their relative abilities. While the truth discovery problem occurs in a wide range of problems related to crowdsourcing and has been of intense focus for the data management community [79], the ability discovery problem gets little attention and is usually treated as a sub-problem: if one knows the truth, it is easier to rank the users based on the choices they make. In turn, if one knows the order of user abilities, it is easier to determine the truth. However, we show in [Section IV](#) that it is not straightforward to rank users correctly, even when given the correct answers to the questions beforehand, which means even the perfect truth discovery method is not guaranteed to perform well for the ability discovery problem. Furthermore, we argue that ability discovery is much more than a subproblem of truth discovery and highlight its importance in two aspects: 1) It has different application scenarios as we discussed in the examples. 2) Different from correctness of answers, user abilities are *abstract and cannot be measured directly*, which makes ability discovery results valuable but also hard to evaluate with *no acknowledged ground truth*.

Assumptions. Similar as in truth discovery, we assume an objective total order on the labels of each item based on their correctness, and a total order on the users based on their latent one-dimensional abilities for choosing the correct labels.

Our approach. We first define an idealized scenario in which the user responses are *consistent* with their abilities across the items and characterize it as the response matrix having the *Consecutive Ones Property* (C1P). We then suggest an efficient spectral method that we call HITSNDIFFS (HND) for reconstructing such ideal orderings if they exist. We prove that in the ideal error-free scenario (better users always make better choices) our method is guaranteed to find the correct ranking, which puts our approach on a stronger theoretical footing as a cross between a heuristic and an exact algorithm. Importantly, our method generalizes to the general non-ideal case and allows us to compare it with existing truth discovery methods for ranking users. One key innovation in our work is the use of Item Response Theory (IRT) [64] to model both label rankings as well as the propensity of users of different abilities to choose such labels, and the previously not made connection to C1P. We utilize 3 different generative models from the IRT literature to generate realistic synthetic data. Experiments on these data show that (i) our new method is *more accurate than existing truth discovery methods*, and (ii) it can also serve as scalable approach that *reconstructs the C1P order if it exists and generalizes much better on non-ideal inputs* than the only other C1P order reconstruction method that works in the non-ideal scenario.

Contributions. (1) We connect the notion of *consistent responses* in heterogeneous multiclass classification to the well-known *Consecutive Ones Property* (C1P) from seriation

theory [4], [22], [29]. We argue that any principled solution to ranking users by their abilities should be able to recover the correct ranking when responses are consistent with abilities.

(2) We propose a novel yet simple adaptation of the HITS algorithm [31] that we call HITSNDIFFS (HND) for ranking users based on their abilities. We prove the surprising result that HND *recovers the consecutive ones ranking* of users when a unique such order exists. Unlike fast combinatorial algorithms for finding the C1P ordering if it exists, HND *can also deal with the general case when no such order exists*. This makes HND an ideal candidate for our problem (and even becomes an exact algorithm in special settings). We compare HND against ABH [4], which is the only other spectral approach that has these properties, and give intuitive and experimental evidence for why HND performs better.

(3) We show how *Item Response Theory (IRT)* [64], which is widely deployed in educational testing, provides a natural and mathematically principled theory (including generative models) for modeling heterogeneous item ranking that includes the C1P as a special case of consistent responses.

(4) We conduct extensive experiments on synthetic datasets generated by 3 polytomous IRT models and show that HND can outperform other existing truth discovery approaches in terms of accuracy of the user ranking. We also show (both in theory and with experiments) that HND has better scalability and accuracy than ABH (which is the only other C1P reconstruction approach known today that can be used for the general ranking problem).

Outline. [Section II](#) defines our problem, draws the connection to the C1P property, and introduces IRT as generalization of C1P. [Section III](#) introduces our approach, gives its formal properties and compares it to closely related work. [Section IV](#) presents experiments. [Section V](#) discusses additional related work on truth discovery before [Section VI](#) concludes. Our code is available online [9].

II. FORMAL SETUP

A. Ability discovery problem formulation

Consider a setting with m users choosing among k options for each of n items. Items are *heterogeneous* in that they have *different options*, as is the case in MCQs used in standardized test settings (see [Figure 1a](#)). This setup is different from typical multiclass classification [12] where all n items have the same class of k labels. To emphasize the difference, we refer to our setup as *heterogeneous multiclass classification*.

User responses can be represented in one-hot encoding as a $(m \times kn)$ *binary response matrix* \mathbf{C} (see [Figure 1b](#)) where each row represents the choices of a user and each column represents an option for some item. The number of non-zeros in \mathbf{C} is mn and the number of non-zeros in each row n .

Our goal is to rank the users by their abilities to choose the “best” options for each item. Each user u_j has a latent one-dimensional *selection ability* θ_j that represents the user’s ability to choose the best of options for each item.

Definition 1 (Ability discovery). *Given m users and their choices among k options³ for n heterogeneous items as binary response matrix \mathbf{C} . Rank the m users by their abilities to correctly choose options for each item.*

Several approaches on homogeneous items assume the probabilities of users getting a correct answer to be identical across questions and encode user abilities as a $(k \times k)$ dimensional confusion matrix per user [12], [79]. In our setting, this is not the case: each option h for item t_i may have a *different* “discrimination score” α_{ih} . In general, the higher α_{ih} for option h is, the more “discriminating” it is (the option’s probability of being chosen more quickly increases with student ability). The probability of answering a question correctly is then a function of the user ability and all the question option discrimination scores. This setup builds upon *Item Response Theory (IRT)* [39], [64], summarized in detail in Section II-D.

Example 3. Figure 1a shows $m=4$ users answering $n=3$ MCQs. Each question has $k=3$ choices labeled A to C in decreasing order of fit. Figure 1b shows an example response matrix \mathbf{C}' and its binary form \mathbf{C} . Assuming that users’ choices are “consistent” with their abilities (i.e. correctness of labeling increases with ability), the only possible ranking of users for the observed \mathbf{C}' is 1, 2, 3, 4, or its reverse. Figure 1c illustrates an IRT model between the latent user abilities and the probability of picking the correct answer when the correct ranking is 1, 2, 3, 4. For example, user u_2 has the ability to label items t_1 and t_2 correctly, thus user u_2 chooses the correct answer A for both items. Our problem is to rank the users by their mastery of the subject based only on the users’ choices without knowing the true labels.

B. The ideal case with consistent responses

We call a response matrix to be “consistent” if there is an unambiguous ordering of users according to their abilities that is reflected in their responses across the items. This way, the ability is a unique skill that is tested across their responses to all items. In this ideal case, if a user u_1 chooses a better option for an item t_1 than user u_2 , then u_1 must also choose an equal or better option for any other item t_2 to reflect this consistency. This implies that there is an implicit ordering among the choices for each item from best to worst and the better users pick better options for every item.

More formally, assume that the user abilities θ_j are all distinct, and also that for every item t_i , the discrimination α_{ih} over all the options are distinct, then there is a *unique linear ordering of the users, and of the options for each item*.

Definition 2 (Consistent Responses). *A response matrix \mathbf{C} is consistent if there exists an assignment of user abilities θ and item discriminations α , s.t. for any pair of users u_1 and u_2 with $\theta_1 > \theta_2$, and for any item t_i where u_1 chooses option h_1 and u_2 chooses option h_2 , we have $\alpha_{ih_1} > \alpha_{ih_2}$.*

³To simplify the discussion and different from Section I, we assume here that each item has exactly k choices. For items with $k' < k$ choices, we can assume them to have $k - k'$ more choices and nobody choosing them.

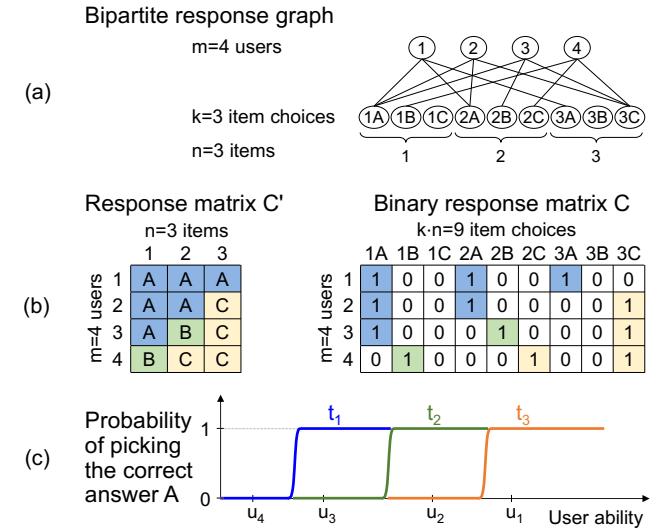


Fig. 1: (a) Ability discovery problem: $m=4$ users choose one from $k=3$ choices of labels A, B, C for each of $n=3$ items. (b) Input: the $(m \times k)$ response matrix \mathbf{C}' , or equally its flattened $(m \times kn)$ binary response matrix \mathbf{C} . (c) Model: the probability of picking the correct answer in terms of the user ability for Items 1,2,3. The abilities of all 4 users are marked on the horizontal axis.

C. Relation to Consecutive ones Property (C1P)

We observe that consistent response matrices, when row-sorted according to user abilities, satisfy a widely studied ordering property in seriation, called the *consecutive ones property (C1P)* [4], [22], [29]. We follow the notation from seriation theory and call it a *P-matrix*.

Definition 3 (C1P, P-matrix & pre-P-matrix [4]). *A binary matrix satisfies C1P and is called a P-matrix if in each column, all the 1’s are consecutive. If the rows of a matrix can be permuted so it becomes a P-matrix, we call it a pre-P-matrix.*

In other words, no 0’s appear between any two 1’s in a column in a P-matrix (see \mathbf{C} in Figure 1b). To see that consistent responses with users sorted by abilities θ give a P-matrix, suppose for a contradiction that a column corresponding to an option for an item has two or more blocks of ones. Then the users corresponding to the zeros in between these blocks will have chosen another option for which the quality is strictly higher or lower than that of this option since the option qualities are assumed to be distinct. But this violates consistency since the rows are ordered by user ability.

Observation 1 (Consistent Responses imply C1P). *A response matrix \mathbf{C} is consistent iff it is a pre-P-matrix.*

Consequently, ranking the users in a scenario of consistent responses corresponds to the problem of determining a permutation of the rows of \mathbf{C} so that the result obeys C1P.

State-of-the-art on C1P. Booth and Leuker [7] (“BL”) proposed the fastest known algorithm for finding all possible permutations of the rows that reconstruct the C1P ordering in time linear in the number of nonzero entries in the matrix,

taking time $O(mn)$ in our setting. However, their method fails to produce an ordering of the rows when the matrix is not a pre-P-matrix, and therefore cannot be used as a general heuristic for simulated or real-world datasets that are not ideal. In contrast, Atkins et al. [4] (“**ABH**”) proposed an elegant spectral method to determine whether a matrix obeys C1P, thus giving a rare continuous method to solve a combinatorial ordering problem. Moreover, it can also be adapted as a heuristic when the input matrix is not a pre-P-matrix. To the best of our knowledge, this is the only currently known method that can be used for ability discovery that is also guaranteed to recover a C1P ranking if it exists.

Our goal. Our goal is to develop a fast and principled algorithm that just like ABH (i) returns a P-matrix in the special case of pre-P matrix inputs, and (ii) can solve the problem in the more general case when the response matrix is *not pre-P*. As we show, the performance of ABH quickly drops in the non-ideal setting (the general IRT setting in [Section II-D](#)) which makes it unusable for ability discovery. We show that our method is more robust and generalizes better, thus being the first method with a useful accuracy for ability discovery that is also guaranteed to solve the ideal case.

D. Relation to Item Response Theory (IRT)

Brief introduction of IRT models. A large body of work from psychological and educational researchers called *Item Response Theory (IRT)* [39], [64] studies the mathematical functions relating the probability of an examinee’s response on a test item to an underlying ability. All major educational tests, such as the Scholastic Aptitude Test (SAT) [39] and Graduate Record Examination (GRE) [30], are based on IRT. IRT forms a mathematically principled, experimentally validated, and widely used theory on how users answer items. [Figure 2](#) summarizes the connections between different IRT models and [Section C](#) contains even more details on IRT.

Binary IRT models can be seen as variations of the standard *logistic* or *sigmoid* function $\sigma : \mathbb{R} \rightarrow [0, 1]$ defined by $\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$, which is widely used in machine learning models and a smooth relaxation of the Heaviside step function $H(x) = \mathbb{I}(x > 0)$ [43]. These models describe the probability $\mathbb{P}_i(\theta)$ for a student with ability θ to answer a question i correctly. Here we only show the 3PL IRT model due to limited space, where θ is the latent ability for each user; a , b , c are latent item factors characterizing the questions and their options, representing discrimination, difficulty and random guessing respectively (Other models can be derived based on [Figure 2](#)):

$$\mathbb{P}_i(\theta) = c_i + (1 - c_i)\sigma(a_i(\theta - b_i)) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta - b_i)}}$$

Multinomial models measure the probability $\mathbb{P}_{ih}(\theta)$ for a student with ability θ to choose an option h for a question i . Thus different from binary models whose parameters belong to questions, multinomial IRT models assign parameters to each option. For example, the Graded Response Model (GRM)

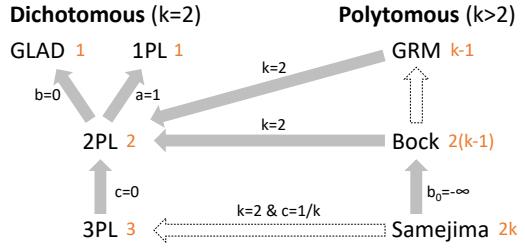


Fig. 2: Correspondences between the discussed IRT models. Orange numbers show number of free parameters per question. Arrows mean “specializes into.” Dashed arrows imply specialization requires special assumptions: Bock to GRM: holds only approximately after fixing $a_h^{\text{Bock}} = h \cdot a^{\text{GRM}}$, Samejima to 3PL: for $k=2$ when $c=1/k$.

model [57] considers a difficulty parameter for each option and a discrimination parameter for each question:

$$\begin{aligned} \mathbb{P}_{ih}(\theta) &= \mathbb{P}_{ih}^*(\theta) - \mathbb{P}_{i,h+1}^*(\theta) \\ \mathbb{P}_{ih}^*(\theta) &= \sigma(a_i(\theta - b_{ih})) = \frac{1}{1 + e^{-a_i(\theta - b_{ih})}} \\ -\infty &= b_{i0} < b_{i1} < \dots < b_{ik-1} < b_{ik} = \infty \end{aligned}$$

The Bock model [6] furthermore assigns a discrimination parameter to each option, and the Samejima model [56] takes into account random guessing by adding a dummy option.

Connection between IRT and C1P. We introduced the definition of consistent responses in [Section II-B](#). When a response matrix is consistent, it is a pre-P-matrix and can be permuted to become a P-matrix which has the consecutive ones property (C1P). If the response matrix is a pre-P-matrix, the corresponding response function of the probability for a user to choose a specific option $h \in \{0, \dots, k-1\}$ can be expressed as the difference between two Heaviside step functions:

$$\mathbb{P}_{ih}(\theta) = H(\theta - b_{ih}) - H(\theta - b_{i,h+1})$$

for appropriately chosen b_{ih} such that:

$$-\infty = b_{i0} < b_{i1} < \dots < b_{ik-1} < b_{ik} = \infty$$

Notice that this response function is exactly the GRM model in the limit of $a \rightarrow \infty$: Whenever the user ability θ is between b_{ih} and $b_{i,h+1}$, then this student chooses option h .

To summarize, IRT models can be seen as a relaxed version of the response function in the ideal case when the response matrix can be permuted to obey C1P. As widely accepted models for MCQs, they strongly support our principle of satisfying the more strict C1P in the ideal case and can be used to generate data in non-ideal cases (see [Section IV](#)).

III. A FAMILY OF HITS ALGORITHMS

We review the HITS algorithm and variants that have been proposed for truth discovery. We then describe a natural averaging version of HITS that we call “AVGHITS.” Our key observation is that the *eigenvector corresponding to the 2nd largest eigenvalue* of its update matrix reconstructs the user (or

row) ordering with C1P if one exists and is unique.⁴ We then describe a variant that we call “HITSNDIFFS” (or HND in short) on a tripartite graph to find this eigenvector efficiently: it keeps an additional vector of differences between adjacent scores and updates it in the loop of the AVGHITS algorithm to compute the ordering of users that we require. We prove that HND returns the correct ordering when the user responses are consistent and compare its time complexity and expected accuracy for non-consistent responses with other methods.

Required background from spectral graph theory. We say that \mathbf{v} is an eigenvector of $(n \times n)$ quadratic matrix \mathbf{A} with eigenvalue λ if $\mathbf{v} \neq 0$ and $\mathbf{Av} = \lambda\mathbf{v}$. If \mathbf{A} is symmetric, then all eigenvalues are real [33]. We use indices to refer to eigenvalues sorted algebraically: $\lambda_1 \geq \lambda_2 \geq \dots \lambda_{n-1} \geq \lambda_n$. We write \mathbf{v}_i to refer to the eigenvector corresponding to eigenvalue λ_i and will be cavalier in referring to it as “the i -th largest eigenvector” when we really mean “the eigenvector corresponding to the i -th largest eigenvalue.”

If the matrix is non-negative and irreducible, then according to the Perron-Frobenius theorem [21], [42], [48] the first eigenvector is also the largest by amplitude ($\lambda_1 = \max_i |\lambda_i|$) and the corresponding eigenvector \mathbf{v}_1 is positive.

A. “HITS” and its variants for truth discovery

Hubs and Authorities (HITS) [31]⁵ is a classic spectral approach that several truth discovery approaches have built upon. The original aim of the algorithm is to rate web pages by two scores: authority and hub score. These scores are recursively defined such that the hub scores are proportional to the sum of the authority scores of the nodes they point to, and the authority scores are proportional to the sum of the scores of the hubs pointing to them, thus reflecting a mutually consistent set of scores [44].

In the context of truth discovery, the authority and hub scores can be interpreted as the user abilities and option correctness scores, respectively. Let \mathbf{C} represent the $m \times n$ binary response matrix where $C_{j,i} = 1$ iff user j chooses item i , and \mathbf{s} be the m -dimensional user score vector, and \mathbf{w} be the n -dimensional item weight vector. In matrix notation, the scores are recursively defined as:

$$\mathbf{s} \leftarrow \beta \mathbf{Cw} \quad \mathbf{w} \leftarrow \alpha \mathbf{C}^\top \mathbf{s}$$

where α and β are normalization constants and \mathbf{C}^\top is the transpose of \mathbf{C} . The algorithm starts from an initial assignment, such as $\mathbf{s} = \mathbf{e}$ and iteratively updates then normalizes \mathbf{w} and \mathbf{s} . The user scores \mathbf{s} will converge to the 1st eigenvector of the matrix \mathbf{CC}^\top .

TruthFinder [73] modifies HITS by first taking *the average instead of the sum* of the chosen item scores as user scores and interpreting them as probabilities of the users being correct on any question. It then defines an item’s score as the probability of it being true given the independent probabilities of all the users choosing the item. When appropriately initialized, the

approach does not require normalization. In the following matrix formulation, let \mathbf{C}^{row} represent the row-normalized response matrix \mathbf{C} :

$$\mathbf{s} \leftarrow \mathbf{C}^{\text{row}} \mathbf{w} \quad \mathbf{w} \leftarrow \mathbf{1} - \exp(\mathbf{C}^\top \log(\mathbf{1} - \mathbf{s}))$$

Investment [47] calculates the ability of users as the sum of the scores of their chosen options, weighted by the user ability they invested in the previous iteration. **PooledInvestment** [47] extends Investment by using a different formula for the item scores. Both variants use non-linear scaling of the item scores with different user-specified hyperparameters.

Our method. We build upon this key idea of updating scores in a bipartite graph of users and items by iteratively summing scores from one side to update the other. However, in contrast to other methods, we focus on the 2nd largest instead of the dominant eigenvector of a new variant and show that this approach is *guaranteed to recover the correct ranking in case of consistent responses*. As we will show in Section IV, no other existing truth discovery method can do that.

B. “AVGHITS”

In our setup, there are nk different choices (k choices for each of n items). Consider a bipartite graph $G = (L \cup R, E)$ that corresponds to the $(m \times nk)$ response matrix \mathbf{C} : Partition L contains a vertex for each of m users: $L = \{u_1, \dots, u_m\}$. Partition R is a collection of n vertex sets $R = \{I_1, \dots, I_n\}$, one for each item. Each set I_i contains $k_i \leq k$ vertices $I_i = \{c_{i1}, \dots, c_{ik_i}\}$ where c_{ih} represents option h of item i . We add an edge to between a user u_j and an option c_{ih} E if user j chooses option h for item i .

To make our derivations easier to follow, we will conveniently assume that each item i has the same number $k_i = k$ of options. Notice however, that this is not required for our approach. We further assume \mathbf{C} to be connected. This requirement applies to all spectral truth ranking methods including HITS as otherwise the relative ranking of users (or items) from different components can’t be established.⁶ Finally, define \mathbf{s} as a $(m \times 1)$ user score vector and \mathbf{w} as a $(kn \times 1)$ option weight vector denoting weights for each of the kn options according to their order in \mathbf{C} .

We call AVGHITS the modification of the HITS update rule that uses *averages* instead of sums to iteratively update the user scores and option weights *in both directions*: the user score s_j is updated to be the average of the weights of all the options that user j picked, and an option weight c_{ih} is updated to be the average of the scores of all users who picked it. In the following matrix formulation, let \mathbf{C}^{row} represent the row-normalized and \mathbf{C}^{col} the column-normalized response matrix \mathbf{C} . At each iteration (until convergence), we update the user score vector \mathbf{s} and the option weight vector \mathbf{w} as follows:

$$\mathbf{s} \leftarrow \mathbf{C}^{\text{row}} \mathbf{w} \quad \mathbf{w} \leftarrow (\mathbf{C}^{\text{col}})^\top \mathbf{s}$$

⁴We consider an ordering and its reverse ordering to be the same.

⁵HITS originally stood for “Hyperlink-Induced Topic Search.”

⁶PageRank achieves the connectivity with the teleport operation.

By combining the above two update equations, we can update user scores between iterations directly by replacing the two normalized response matrices with one *update matrix* \mathbf{U} :

$$\mathbf{s} \leftarrow \underbrace{\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}}_{\mathbf{U}} \mathbf{s} \quad (1)$$

These iterations are not yet very helpful. Indeed, we observe that the largest eigenvector of \mathbf{U} is the all-ones vector \mathbf{e} , and this is the vector of user scores that AVGHITS converges to. It turns out that it is the eigenvector corresponding to the *2nd largest eigenvalue* of \mathbf{U} that we seek.

C. Our algorithm “HITSNDIFFS” (HND)

In the following, we show a simple algorithm to find the 2nd largest eigenvector ordering of \mathbf{U} and prove that it can be used to find the unique consecutive ones ordering of the response matrix \mathbf{C} . By “the eigenvector ordering”, we mean the ranking of entries in this eigenvector in terms of their values. For example, $\mathbf{v}_1 = \{0.36, 0.8, 0.48\}$ and $\mathbf{v}_2 = \{0.48, 0.64, 0.6\}$ have the identical ordering $\{3, 1, 2\}$ or its reverse $\{1, 3, 2\}$. Our algorithm does not return the 2nd largest eigenvector of \mathbf{U} but instead returns a vector with the *identical ordering*.

The 2nd largest eigenvector of a matrix can be found using a variant of the deflation method [41], [43], which we will discuss in detail in Section III-F. Here we present a novel, conceptually simple, and faster algorithm that we term “HITS and DIFFS” (HITSNDIFFS or HND) that extends AVGHITS from a bipartite to a tripartite graph and whose iterative updates converge to a user ranking that is *guaranteed to be C1P in the ideal case*, and that *performs well also in more general settings*. This approach leverages particular properties of our problem that don’t apply to the 2nd largest eigenvector orderings of any matrix from more general settings.

First, we propose a new intermediate step that calculates differences between user scores in the iterative updates of AVGHITS. Rather than updating the user scores iteratively, HITSNDIFFS updates the *differences* between adjacent user scores by using a suitably modified update matrix, and results in the scores converging to the ordering according to the second largest eigenvector of \mathbf{U} . Furthermore, this modification only adds a linear overhead of computing the user score difference vectors and normalizing it in every iteration. As we will show in Theorem 2, when the response matrix obeys C1P, then HND reconstructs the correct ordering of the rows.

As shown in Figure 3, we define a new vector \mathbf{s}^{diff} of differences in user scores with entries $s_j^{\text{diff}} = s_{j+1} - s_j$, $j \in [m-1]$. This is equal to $\mathbf{s}^{\text{diff}} = \mathbf{S}\mathbf{s}$ where $\mathbf{S} \in \mathbb{R}^{(m-1) \times m}$ is shown in Figure 3. In the reverse direction, there are infinitely many vectors \mathbf{s} that can be generated from a given \mathbf{s}^{diff} , all shifted by different constants. Since we only want a final ordering of users, we can WLOG set the first element of the vector \mathbf{s} to be 0. The transformation then is $\mathbf{s} = \mathbf{T}\mathbf{s}^{\text{diff}}$ where $\mathbf{T} \in \mathbb{R}^{m \times (m-1)}$ is the lower unit triangular matrix⁷.

⁷It is this fixing that intuitively keeps the ordering, but changes the actual amplitudes.

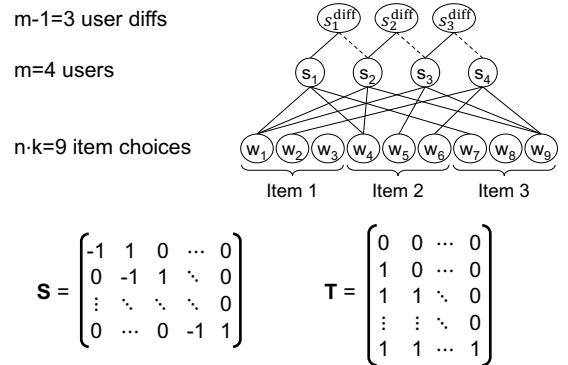


Fig. 3: HITSNDIFFS uses a *3-partite graph* of option weights, user scores, and user diffs. Contrast this graph with Figure 1. The update equations (see Algorithm 1) use two re-shaping matrices \mathbf{S} and \mathbf{T} .

We can now get a user difference score update rule:

$$\mathbf{s}^{\text{diff}} \leftarrow \mathbf{S}\mathbf{s} = \mathbf{S}\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}\mathbf{s} = \underbrace{\mathbf{S}\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}}_{\mathbf{U}^{\text{diff}}} \mathbf{T}\mathbf{s}^{\text{diff}} \quad (2)$$

In other words, $\mathbf{U}^{\text{diff}} = \mathbf{S}\mathbf{U}\mathbf{T}$ is a “difference update” matrix that is used to update \mathbf{s}^{diff} from one iteration to the next. With these update equations, \mathbf{s}^{diff} converges to the largest eigenvector of \mathbf{U}^{diff} . Our algorithm HND that implements this is described in Algorithm 1.

We now prove the connection between the 1st eigenvector of \mathbf{U}^{diff} and the 2nd largest eigenvector of \mathbf{U} .

Lemma 1 (Eigenvector correspondence). \mathbf{x} is the 2nd largest eigenvector of \mathbf{U} iff $\mathbf{y} = \mathbf{S}\mathbf{x}$ is the largest eigenvector of \mathbf{U}^{diff} .

Proof sketch. Due to the limit of space, we only provide the high-level ideas of our proofs in the paper. First, we can find out that each row of \mathbf{U} has sum 1. Using this, we can prove that the largest eigenvector of \mathbf{U} is in the direction of the all ones vector $\mathbf{e} = \mathbf{1}_m$ if the largest eigenvalue of \mathbf{U} has multiplicity 1 (i.e. the graph is a single connected component). Let \mathbf{x} be an eigenvector of \mathbf{U} that is not in the direction of the all ones vector, i.e. $\mathbf{x} \neq \alpha\mathbf{e}$. Note that $\mathbf{TS} = (\mathbf{I}_m - \mathbf{ee}_1^{\top})$ and each row of \mathbf{SU} sums to 0. Then,

$$\begin{aligned} \mathbf{Ux} &= \lambda\mathbf{x} \\ \mathbf{SUx} &= \lambda\mathbf{Sx} \\ \mathbf{SU}(\mathbf{I}_m - \mathbf{ee}_1^{\top})\mathbf{x} &= \lambda\mathbf{Sx} \\ \mathbf{SUTSx} &= \lambda\mathbf{Sx} \\ \mathbf{U}^{\text{diff}}\mathbf{y} &= \lambda\mathbf{y}, \quad \text{where } \mathbf{y} = \mathbf{Sx} \end{aligned} \quad (3)$$

Therefore, \mathbf{U}^{diff} has exactly the same eigenvalues as \mathbf{U} except the largest eigenvalue 1, and the eigenvectors of \mathbf{U}^{diff} are the differences between the entries of the corresponding eigenvector of \mathbf{U} . Thus we prove the lemma. \square

Theorem 1 (2nd eigenvector of AVGHITS recovers C1P). If \mathbf{C} is a pre-P-matrix with a unique consecutive ones ordering of its rows and each row has the same row sum, then this

Algorithm 1: HITSNDIFFS (HND-power): A fast implementation of equation (2) to calculate the 2nd eigenvector ordering of $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}$

Input: Response matrix \mathbf{C} , randomly initialized user scores \mathbf{s}_0
Output: User scores \mathbf{s}

```

1:  $\mathbf{s}^{\text{diff}} \leftarrow \mathbf{s}_0^{\text{diff}}$  // initialize user score differences
2: repeat
3:    $\mathbf{s} \leftarrow \mathbf{T}\mathbf{s}^{\text{diff}}$  // update user scores
4:    $\mathbf{w} \leftarrow (\mathbf{C}^{\text{col}})^{\top}\mathbf{s}$  // update option weights
5:    $\mathbf{s} \leftarrow \mathbf{C}^{\text{row}}\mathbf{w}$  // update user scores
6:    $\mathbf{s}^{\text{diff}} \leftarrow \mathbf{S}\mathbf{s}$  // update user score differences
7:   Normalize  $\mathbf{s}^{\text{diff}}$  to be a unit vector
8: until convergence or iteration limit
9:  $\mathbf{s} \leftarrow \mathbf{T}\mathbf{s}^{\text{diff}}$ 

```

ordering of the rows of \mathbf{C} is given by the ranking of the rows sorted by values in the 2nd largest eigenvector of \mathbf{U} .

Proof sketch. We can first prove that if \mathbf{C} is a pre-P-matrix with a unique consecutive ones ordering of its rows and each row has the same row sum, \mathbf{U} is an R-matrix (defined in [4]) where in each row and column, the entries closer to the diagonal are larger than or equal to the further entries. Using this, we can prove every entry in \mathbf{U}^{diff} is non-negative by computing each entry in \mathbf{U}^{diff} step by step according to its definition, which means \mathbf{U}^{diff} is a non-negative matrix. We can now apply the Perron-Frobenius Theorem [21], [48]: there exists a non-negative eigenvector of \mathbf{U}^{diff} corresponding to the largest eigenvalue of \mathbf{U}^{diff} . We know \mathbf{U}^{diff} has exactly the same eigenvalues as \mathbf{U} , except the largest eigenvalue 1, and the eigenvectors of \mathbf{U}^{diff} are the differences between the elements of the corresponding eigenvector of \mathbf{U} . Since the differences between the elements of the eigenvector corresponding to the 2nd largest eigenvalue of \mathbf{U} (largest eigenvalue of \mathbf{U}^{diff}) is non-negative, that eigenvector of \mathbf{U} is monotonic. Therefore, sorting the rows according to the second largest eigenvector ordering of the corresponding update matrix \mathbf{U} gives a P-matrix, proving the theorem. \square

Theorem 2. If \mathbf{C} is a pre-P-matrix with a unique C1P ordering of its rows and each row has the same row sum, then HND reconstructs the consistent ordering of the users taking linear time in the number of nonzeros in \mathbf{U} per iteration.

Proof sketch. From Lemma 1, we know by converting the converged largest eigenvector of \mathbf{U}^{diff} back into a user score, we regain the ordering of the rows according to values in the second largest eigenvector of \mathbf{U} . This, along with Theorem 1, proves this theorem that HND detailed in Algorithm 1 reconstructs the ideal consistent ordering. \square

D. Decile entropy-based symmetry breaking

Notice that reversing the order of a P-matrix still leaves it as a P-matrix. Thus all methods for solving C1P suffer from a natural *symmetry breaking problem*: they have to decide between the order returned by an algorithm or its exact inverse.

Our solution to this symmetry-breaking problem is motivated by the following observation: *users with higher ability tend to converge on the correct option as a majority answer*, while users with lower ability at the other end of the ordering tend to answer randomly. This idea is similar to the main argument in [35] that experts tend to answer similar correct answers. Thus the lower end of the user ordering has a *higher entropy* in the choices picked than the higher quality end. Notice that this idea is also implicit in IRT models with random guessing where users with low ability choose uniformly random among the options (hence have high entropy), whereas users of high ability pick the single correct choice.

We operationalize this idea in a new heuristic that is very effective in practice: Given a ranking of the users, we compute, for the top and the bottom user decile, the average entropy of the chosen item options across all items. We pick the side with lower entropy as the users with higher quality. We use this “*decile entropy method*” for both HND and ABH in our experiments.

E. Why HND works better than ABH

HND and ABH rely on strikingly similar intuitions about spectral properties of matrices: HND ranks users by the 2nd largest eigenvector of \mathbf{U} (whose difference is the largest eigenvector of $\mathbf{U}^{\text{diff}} = \mathbf{SC}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}\mathbf{T}$), whereas it can be shown that ABH ranks users by the 2nd smallest eigenvector of the Laplacian matrix \mathbf{L} of \mathbf{CC}^{\top} (whose difference is the smallest eigenvector of $\mathbf{M} = \mathbf{SLT}$). In an ideal scenario with consistent responses (thus in an IRT scenario with very large discrimination scores), both methods are guaranteed to return the correct C1P ordering.⁸

We can also expect the accuracy to be identical in the other extreme scenario where all questions have 0 discrimination. But how can we expect their accuracy to compare in the more general scenario?

We interpret the general IRT scenario as random perturbations [61] from the ideal C1P case. Now notice that the smallest eigenvector of \mathbf{M} is identical to the largest eigenvector of $\beta\mathbf{I}_{m-1} - \mathbf{M}$ where β is larger than all the entries and all the eigenvalues of \mathbf{M} .⁹ Thus the comparison of HND and ABH corresponds to the largest eigenvector of \mathbf{U}^{diff} against $\beta\mathbf{I} - \mathbf{M}$. Since both matrices have all non-negative entries in the ideal scenario, we know from the Perron-Frobenius theorem [21], [48] that all values in their largest eigenvector are non-negative.

The user score of the k th user equals to the cumulative sum of the first $k-1$ entries in the eigenvector. In the ideal case when \mathbf{C} is a C1P matrix, every entry of \mathbf{s}^{diff} is non-negative so \mathbf{s} give us a perfect ranking of the students. In the non-ideal scenario when \mathbf{C} is not a C1P matrix and the users

⁸Recall that they are guaranteed to return the same ordering, but not the same eigenvector.

⁹To see that, assume $\mathbf{Av} = \lambda\mathbf{v}$. Then $(\mathbf{A} + \beta\mathbf{I})\mathbf{v} = \mathbf{Av} + \beta\mathbf{I}\mathbf{v} = \lambda\mathbf{v} + \beta\mathbf{v} = (\lambda + \beta)\mathbf{v}$. Thus if \mathbf{v} is an eigenvector of \mathbf{A} with eigenvalue λ , then \mathbf{v} is also an eigenvector of the spectrally shifted matrix $\mathbf{A} + \beta\mathbf{I}$, but with eigenvalue $\lambda + \beta$.

are permuted by their abilities, the sign of the entries in \mathbf{s}^{diff} change. When the eigenvector is even, a simple sign change in one of the entries does not influence the entire ranking of \mathbf{s} but when the eigenvector has a large variance, a simple sign change in one large entry can break the entire ranking. For example, if the k th entry of \mathbf{s}^{diff} is quite large but the sign is negative, the error of the ranking of the $(k+1)$ th students can be very large.

Based on our observation above, we expect HND to work better than ABH as the variance of largest eigenvector of $\beta\mathbf{I} - \mathbf{M}$ should be much larger compared to \mathbf{U}^{diff} . The result is verified with dedicated experiments in [Section IV-D](#): [Figure 6a](#) shows our observations on the variances of the \mathbf{s}^{diff} for $\beta\mathbf{I} - \mathbf{M}$ and \mathbf{U}^{diff} . [Figure 6b](#) and [Figure 6c](#) verify that ABH is less accurate and less stable than HND.

F. Complexity Comparison

We analyze the asymptotic time complexity of the various methods. We compare HND against (i) existing C1P reconstruction algorithms BL and ABH, and (ii) the deflation method [41], [43] as an alternative method to compute the 2nd largest eigenvector for AVGHITS.

The time complexity depends on the number of users m , the number of questions n , and the number of iterations t which may be different for different methods.

We assume $t \ll n$ and $t \ll m$ and thus only focus on m and n . Notice that although the response matrix \mathbf{C} is a $(m \times kn)$ -matrix, it has only $\mathcal{O}(mn)$ non-zero entries since every user can pick only one label per question.

HITSNDIFFS. A naive way to calculate the ranking is to first compute \mathbf{U}^{diff} and then use the power method on it. However, computing \mathbf{U}^{diff} requires a matrix-matrix multiplications before the iterations with time complexity $\mathcal{O}(m^2n)$. Since \mathbf{U}^{diff} is a $(m-1) \times (m-1)$ matrix, the time complexity to run the power method on \mathbf{U}^{diff} is $\mathcal{O}(m^2)$ per iteration. This gives a total time complexity for the naive implementation as $\mathcal{O}(m^2n) + \mathcal{O}(m^2t) = \mathcal{O}(m^2n)$. By instead running the mutual updates of \mathbf{w} , \mathbf{s} and \mathbf{s}^{diff} as described in [Algorithm 1](#), we can replace matrix-matrix multiplications with several matrix-vector multiplications and thereby get a more efficient implementation of HITSNDIFFS in $\mathcal{O}(mnt)$. In other words, the speed-up results from applying the associativity law and replacing the calculation $\mathbf{s} \leftarrow (\mathbf{SC}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top} \mathbf{T})\mathbf{s}^{\text{diff}}$ with $\mathbf{s} \leftarrow \mathbf{S}(\mathbf{C}^{\text{row}}((\mathbf{C}^{\text{col}})^{\top} (\mathbf{T}\mathbf{s}^{\text{diff}})))$. One detail is that we need however to implement [Line 3](#) differently. Materializing the matrix \mathbf{T} would take $\mathcal{O}(m^2)$. Instead, we calculate the entries for \mathbf{s} from \mathbf{s}^{diff} via cumulative summation (e.g. via `numpy.cumsum` in Python).

The deflation method. [Theorem 1](#) showed that our problem can be formulated as finding the 2nd largest eigenvector \mathbf{v}_2 of \mathbf{U} . The problem of finding the eigenvector corresponding to the second largest eigenvalue of a given matrix \mathbf{A} can be solved with the *deflation method* [41], [43]. The idea is to first calculate the dominant eigenvector \mathbf{v}_1 , and then eliminate the influence of the \mathbf{v}_1 from \mathbf{A} to get a new matrix \mathbf{B} whose 1st eigenvector is the 2nd eigenvector of \mathbf{A} . Then \mathbf{v}_2 of \mathbf{A} can

be obtained by using the power method on \mathbf{B} . We next argue (and later show experimentally in [Section IV-C](#)) that using the deflation method is slightly less efficient than HND (in addition to being not as simple to formulate as HND-power).

The most widely known deflation method [41], [43]

only works for symmetric matrices and does not apply to the asymmetric \mathbf{U} . [68] presents several more variants of the deflation method including some that work for non-symmetric matrices. Most of those methods either require matrix-matrix multiplication or both the left dominant eigenvector and the right dominant eigenvector. The only exception is Wilkinson's vector annihilation [70] which only needs the right dominant eigenvector (which we know is a unit vector in the direction of the all ones vector in our case). However, [68] claimed that this method is difficult to apply in practice because of the need to conduct annihilation between the power iterations and we found no open-source implementation.

For our experiments in [Section IV](#) we implement *Hotelling's matrix deflation* [69] which uses both the left and right largest eigenvectors and thus requires one more round of the power iteration.¹⁰ The experimental result in [Section IV-C](#) verifies that HND is not just conceptually simpler but also slightly more efficient than the deflation method.

ABH [4]. To reconstruct the C1P ordering, ABH requires the computation of the Fiedler vector [18], which is the eigenvector corresponding to the 2nd smallest eigenvalue of the related Laplacian matrix \mathbf{L} .

The original ABH paper [4] does not propose an explicit solution and instead refers to the Lanczos algorithm [32], [46], whose time complexity is $\mathcal{O}(dmt)$ with d being the average number of non-zero entries in a row of a given $(m \times m)$ matrix \mathbf{A} . When the Laplacian matrix is dense as in our scenarios the time complexity of the Lanczos algorithm is $\mathcal{O}(m^2t)$. It is efficient for eigenvector computations on large symmetric matrices [10], and the state-of-the-art Fiedler vector solver [26] uses Lanczos.

However, implementation by libraries such as Scipy [66] and Tenpy [24], require the full matrix as input, which would require us to compute the Laplacian matrix first. This calculation involves matrix-matrix multiplications and, as we show in [Section IV-C](#), results in $\mathcal{O}(m^2n)$.

We provide another solution for ABH which is not in the original paper [4]. Similar to how we implement HND as [Algorithm 1](#), we can also implement ABH by using the power method on the matrix $\beta\mathbf{I}_{m-1} - \mathbf{M}$ to get its largest eigenvector without having matrix-matrix multiplications. As we discussed in [Section III-E](#), this largest eigenvector of $\beta\mathbf{I}_{m-1} - \mathbf{M}$ is identical to the smallest eigenvector of \mathbf{M} which can be used to compute the order of the 2nd smallest eigenvector of \mathbf{L} in the same way as [Algorithm 1](#). The total time complexity for this algorithm is $\mathcal{O}(mnt + m^2t)$. When m and n are close or $m < n$, the time complexity becomes $\mathcal{O}(mnt)$ which is the

¹⁰We first calculate the dominant left eigenvector via power iteration, then deflate the matrix, and then calculate the dominant right eigenvector on the deflated matrix.

same as HND. However, when $m \gg n$, the time complexity becomes $\mathcal{O}(m^2t)$, which is larger than $\mathcal{O}(mnt)$ of HND.

BL [7]. The original paper by Booth and Leuker (BL) [7] for reconstructing the C1P property can work directly on the initial response matrix and runs in $\mathcal{O}(mk + n + f)$, where f is the non-zero entries in the matrix. In our setup where $f = \mathcal{O}(mn)$, the time complexity is $\mathcal{O}(mn)$. Therefore, BL is the fastest method when it works. Since it cannot be used for solving ability discovery in general, we are not using it in our experiments.

HITS [31], Truthfinder [73], Investment [47], PooledInvestment [47]. All these methods are iteration-based variants of HITS that take at least $\mathcal{O}(mnt)$ and differ in how they iterate between the user and the item scores. In practice, only HITS can be defined as an eigenvector problem with a closed-form solution and efficient linear algebra implementation. Truthfinder converges in practice, while Investment and PooledInvestment can not converge and return different results depending on initialization. Our approach in contrast comes with the same computational properties as HITS: guarantee of convergence, unique solution, an intuitive formulation as a spectral problem, and an efficient matrix implementation.

IV. EXPERIMENTS

Our experiments compare the *accuracy* of ability discovery and the *scalability* of the various methods. The main take-aways from the experiments are: 1) HND robustly returns rankings for users with accuracy on average *better than or equal to* other truth discovery methods; 2) HND is competitive with two “*cheating competitors*” (that are provided the ground truth information about the correct options for each question that is usually not available); 3) HND has better scalability as a C1P reconstruction algorithm than ABH.

A. Experimental setup

Environment. All scalability experiments are run on Intel Xeon E5-2680 CPUs with an exclusive environment and 128G allocated memory. We implemented HND in Python 3.8.1.

Benchmarks. [59] provides an extensive benchmark for the *truth discovery problem*, and [79] provides a broad survey of existing truth discovery methods. Two points stand out: 1) All open-source datasets used in the two papers *lack ground truth for user abilities*. 2) All 20 datasets fall under the setting with *homogeneous items*. As we discussed in Section I, it is easy to understand the lack of ground truth for the ability discovery problem because the *user abilities are abstract and not able to be obtained from external knowledge*. To make up for it, we first create synthetic data based on the polytomous models from *Item Response Theory (IRT)* (recall Section II-D). Moreover, we use a real-world MCQ dataset with approximate (but not accurate enough) ground truth as a supplementary evidence to verify the usefulness of HND in Section IV-E.

Polytomous synthetic data generator. We use the three polytomous IRT models from Section II-D (GRM [57], Bock [6] and Samejima [56]) to generate synthetic data sets with known ground truth. Samejima model takes random

guessing into account so it models the educational test scenario where students try to maximize their scores. Bock and GRM models with no random guessing models the crowdsourcing scenario where workers usually do not guess.

By default, we set user ability θ to be within $[0, 1]$, item difficulty b to be within $[-0.5, 0.5]$, and the item discrimination a to be within $[0, 10]$, all uniformly random. Besides varying the number of users, items and options, Section IV-B also has experiments with shifted b 's (chosen to achieve a certain percentage of users giving correct answers).¹¹

Methods and their implementations. For a thorough evaluation, we created three alternative implementations of HND and two alternative implementations of ABH. **HND-power** follows Algorithm 1 which only involves matrix-vector multiplications. **ABH-power** is our novel kimplementation of ABH that avoids matrix-matrix multiplications by using the power method on the matrix $\beta\mathbf{I}_{m-1} - \mathbf{M}$. **ABH-direct** is the implementation of ABH using the Lanczos algorithm as suggested by [4]. Section III-F discussed the drawback of requiring matrix-matrix multiplications. We use an efficient sparse Linear Algebra Python library called Scipy [66]. **HND-direct** implements HND similarly by directly computing the 2nd largest eigenvector of \mathbf{U} by using the Arnoldi algorithm [3], which can be considered the general version of the Lanczos algorithm on asymmetric matrices, also using Scipy. **HND-deflation** implements HND with the deflation method discussed also in Section III-F. For HND-power, ABH-power and HND-deflation, the criterion for convergence is a maximal L2-norm of 10^{-5} over the change. For our experiments (other than Section IV-C) we used “HND-power” for HND and “ABH-direct” for ABH since they turned out to be the fastest implementations.

We also implemented **HITS** [31], **TruthFinder** [73], **Investment** and **PooledInvestment** [47]. Since none of those iteration-based approaches (except HITS) allows an efficient matrix formulation, our implementation in Python uses loops and is not efficient. We thus do not report scalability experiments on those methods as native implementation in C++ would bring those close to HND as discussed in Section III-F. For Investment and PooledInvestment (which do not converge) we use 10 iterations instead of tuning the number of iterations.

Two cheating baselines. To show the effectiveness of HND, we also compare HND with two “*cheating competitors*” that are given additional ground truth information about questions that is usually not available: **True-answer** has information about which choices are correct for each question (which is usually not known in our scenario) and then ranks users by the number of correctly answered questions. **GRM-estimator** uses a Python package called GIRTH [58] that estimates the parameters of a GRM model including user abilities. However, it *requires knowing the order of options* for each question by correctness.

¹¹For experiments with GRM data, we use the data generator from the GIRTH package which requires at least $k = 3$ options. We implemented Bock and Samejima generators ourselves and thus options can start from $k = 2$.

Our comparison with this approach is notable because it is the theoretically “best” model to fit data generated by the same synthetic GRM process.

B. Accuracy on synthetic data

Accuracy. To determine the accuracy of a method, we calculate *Spearman’s rank correlation coefficient* [60] between the returned user ranking and the ground truth ranking by their actual abilities. Spearman’s correlation is defined as the *Pearson correlation between the rankings* of two scoring functions and ranges between -1 and 1 . It is similar to Kendall’s correlation, yet strictly preferred if there are ties in the data [49]. There can be negative accuracy at times (not shown in Figure 4), which means the returned ranking is negatively correlated or random whose coefficient is near 0 .

Setup. We conduct experiments to determine the accuracy as function of the 1) *number of items* n , 2) *number of users* m , 3) *number of options* k , 4) *option difficulties* b_{ih} , and 5) *probability of questions to be answered* p . In the first experiment, we use data generated according to the three polytomous models (GRM, Bock, Samejima) from Section II-D to also show the robust performance of HND for all three models. In other experiments, we only use data generated according to the Samejima model since it is the most general one to avoid redundancy. To verify the ability of algorithms to recover a C1P ranking, we also generate data that 6) *follows the consistent response property* (which as discussed in Section II-C is the case for IRT models when the discrimination $a_{ih} \rightarrow \infty$). Every question has the same number of options, and every user answers every question. By default, we set the users $m = 100$, items $n = 100$, and options $k = 3$.

1. Varying number of questions n (Figures 4a to 4c¹²)

HND has better than or equal accuracy as the other methods even including the two cheating competitors over data generated by all three models. Notice that the GRM-estimator works poorly for Samejima because it does not take random guessing into account.

2. Varying number of users m (Figure 4d). HND works here also better than or equal to other approaches (except for the data point with low m where the cheating competitors win).

3. Varying number of options k (Figure 4e). HND stays top and accurate, even slightly outperforming True-answer.

4. Varying question difficulties b_{ih} (Figure 4f). Here, we change the difficulty range from the default $[-0.5, 0.5]$ to 7 different ranges, $[-1, 0]$, $[-0.75, 0.25]$, $[-0.5, 0.5]$, $[-0.25, 0.75]$, $[0, 1]$, $[0.25, 1.25]$, $[0.5, 1.5]$, while user abilities θ_j remain at $[0, 1]$. Thus even the least able user has a high probability to answer a difficult question in the easiest setting, while even the best user can be incorrect for some easy questions in the hardest setting. The x-axis here is the *average accuracy* on the questions across all the users. In all scenarios, we see HND outperforms other competitors.

5. Varying probability p of answering a question (Figure 4g). To show that HND works for more general scenarios

where users *answer different number of questions*, we vary the probability of questions to be answered. For each pair of question and user, there is the probability of p for the user to answer the question. We see that HND performs well even when the dataset is not complete.

6. C1P (Figure 4h).¹³ In addition to the three multinomial IRT models, we also generate response matrices that are consistent and can be reconstructed to be a P-matrix. These responses correspond to a random GRM instance with very strong discrimination a . We use these matrices to verify the effectiveness of HND in reconstructing a C1P permutation. To avoid ties in the rankings and provide a unique C1P ordering, we set both the user ability θ and the difficulty parameter b to be within $[0, 1]$, randomly chosen. We see that HND and ABH are indeed the only two methods that can reconstruct the C1P permutation if there exists one.

Summary. HITSNDIFFS is a *robust* method that outperforms the other approaches in most setups, especially those with high discrimination. We see this as vindication for designing an approach based on the principle that *consistent answers need to be solved correctly*. HND is also competitive even against the two cheating approaches which have the best item answer given (i.e. they have access to an oracle that can solve the entire problem of truth discovery). Moreover, we verified that HND and ABH are indeed the only ones that can reconstruct a C1P permutation if it exists.

C. Scalability experiments

Figures 5a and 5b show the scalability of our various implementations of ABH and HND, as well as the GRM-estimator w.r.t.

number of users (m) and questions (n). Each shown data point is the median over 5 runs, and we set a timeout of 1,000 seconds. **HND vs. ABH.** Figure 5a shows that ABH-direct and HND-direct scale with $O(m^2k)$ in the number of users as predicted in Section III-F. The theoretic time complexity of ABH-power is $O(m^2t)$ when m is much larger than n , and thus it also takes quadratic time. In contrast, HND-power can scale linearly and is about 20% faster than HND-deflation on average for $m > 1000$ users as it needs only one round of the power method. Figure 5b shows that although ABH-direct is slightly faster for fixed few users, all implementations are efficient even for a large number of questions.

GRM-estimator. As a representative of max likelihood parameter estimation, the GRM-estimator is expected to perform best on GRM data. However, Figure 5 shows that such parameter estimation is by orders of magnitude slower than HND.

Summary. HND scales asymptotically and practically better than the other existing C1P reconstruction algorithm ABH in the number of users. Moreover, our intuitive Algorithm 1 is slightly faster than an adaptation of the deflation method.

¹²The GRM estimator does not work when the question number is large.

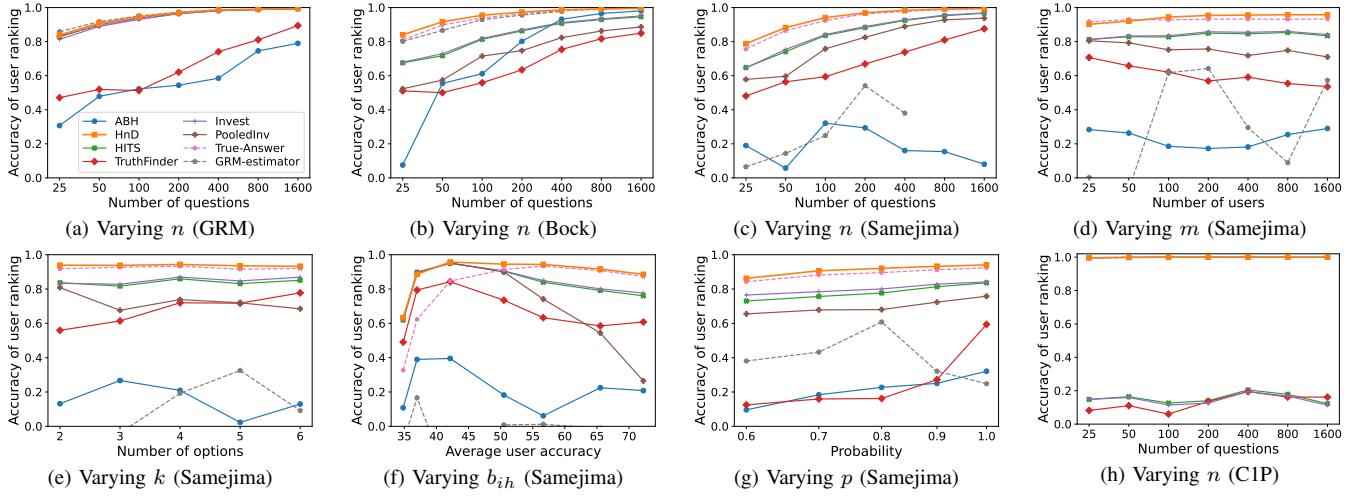


Fig. 4: Section IV-B: Results of accuracy experiments (the legend is in the first figure).

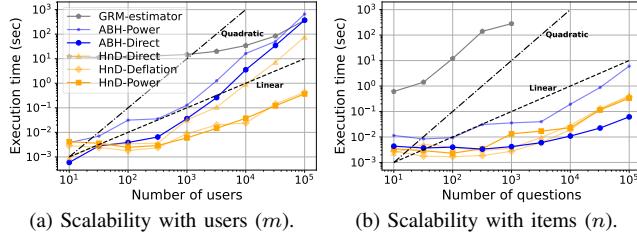


Fig. 5: Section IV-C: Scalability experiments with $n = 100$ items and increasing numbers of users m in (a), or $m = 100$ users and increasing numbers of items n in (b). The experiments confirm that our method (HnD) scales linearly in the number of items and users, whereas ABH (even trying various alternative methods) has an unavoidable quadratic scalability in the number of users.

D. Stability experiments for ABH and HND

We next experimentally verify our prediction from Section III-E that HND generalizes better from the ideal case than ABH. In this setup, we fix $m = 100$ users, $n = 100$ items, $k = 3$ options, with user abilities and item difficulties equally spaced between $[0, 1]$ and $[-0.5, 0.5]$ respectively. For one item, all the option difficulties are the same. All items have identical discrimination a and all the options in one question have equally spaced a (as in the GRM model).

We then vary the question discrimination scores and compare the (i) variance of the respective eigenvectors used for ranking; (ii) the normalized average difference in rank between each user's ranking,¹⁴ and (iii) the average accuracy of the predicted rankings for HND and ABH across repeatedly sampled response matrices.

Figure 6a shows our observation from Section III-E that the variance of the largest eigenvector of \mathbf{U}^{diff} of HND is much smaller than $\beta \mathbf{I}_{m-1} - \mathbf{M}$, which is expected to lead to the

¹³In the experiments, PooledInv returned all negative coefficients but we consider its ranking to be the reverse one.

¹⁴Here difference means the average difference of each user's rankings from different runs, scaled down to $[0, 1]$ by the user number.

better stability and accuracy of the HND rankings. Figure 6b confirms that the ranking of a user is more stable for HND. Figure 6c shows the resulting increase in the accuracy of HND over ABH. This confirms our original goal to develop a spectral method that can achieve the same C1P ranking as ABH, yet generalizing better in the non-ideal case.

E. Accuracy experiments on real-world data

As mentioned in Section IV-A, we do not know any existing benchmark with a known true ranking of users by their abilities. In order to still verify the performance of HND on real-world datasets we use the ranking of the “True-answer” baseline as the ground truth. Notice that although this baseline performs well in our synthetic experiments, it is far from the perfect gold standard (sometimes even outperformed by HND) so the experimental result in this subsection should be seen only as a supplementary evidence. The six used real world MCQ datasets are from [35].

Figure 7 shows the average experimental result of six datasets where PooledInvestment and HITS perform slightly better than HND. However, we need to emphasize several points: (1) All datasets are very small in terms of question numbers (from 20 to 36) but have double user numbers on average, which indicates their limited discrimination. (2) There is no consistent winner on all six datasets (see detailed result in Section D), an observation also made by [79] for the related truth discovery problem. (3) All other models except ABH with poor performance tend to have more similar accuracy than HND while HND tops them by far on 2 of the 6 datasets, which shows the novelty of HND and its usefulness on data of different distributions.

V. ADDITIONAL RELATED WORK

In this section, we discuss additional approaches for the truth discovery problem. This is in addition to existing C1P reconstruction algorithms discussed in Section II-C and HITS-based truth discovery approaches discussed in Section III-A.

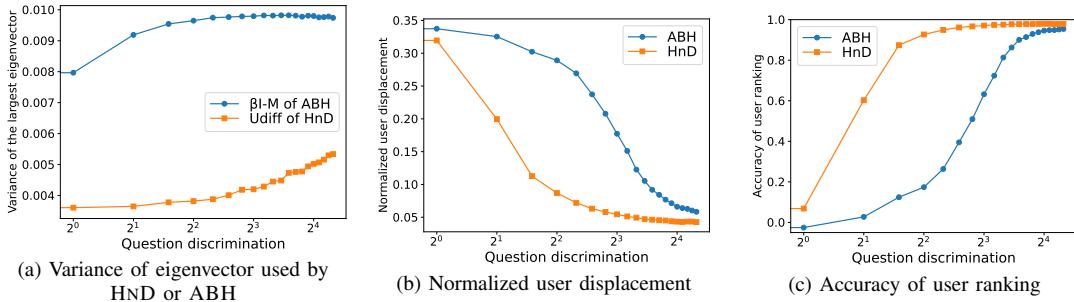


Fig. 6: Section IV-D: Stability experiments: (a) The variance of the eigenvector used by HnD is smaller than that used by ABH, which makes it more robust to perturbations from the ideal C1P case. This leads to HnD having lower difference in the user ranking (b) and higher accuracy (c).

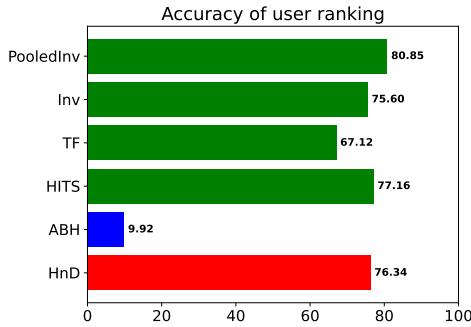


Fig. 7: Section IV-E: Correlation of user ranking on real-world datasets with the “True-answer” baseline that serves us as approximate gold standard user ranking. Notice that the true ranking of users by ability is not known.

Spectral approaches. Dalvi et al. [11] proposed two methods that output the user abilities and item labels with the help of eigenvector computation. Ghosh et al. [23] proposed a method that only outputs the item labels, which involves calculating the first eigenvector of a symmetric matrix. Both approaches work only for binary problems and are not obvious to generalize for $k > 2$ options.

Other truth discovery approaches. [35] proposes the concept of experts and utilizes the observation that experts are more likely to reach consensus on a set of single questions (called hyper-questions in the paper) to conduct majority vote on hyper-questions instead of single questions but cannot quantify user abilities nor rank them. [40] relies on embeddings that cannot be easily converted into a ranking on users. [36] uses confidence and focus on long-tail data. [12], [28] are optimization-based methods that only consider homogeneous questions (recall Section II-A).

Other truth discovery problems. Many approaches have been proposed for truth discovery. Most have different setups and are not applicable to our problem. [62], [72] change the setup of the problem by assigning different tasks to two groups of workers, where the first group answers the questions and the second group evaluates the answers. [16], [52], [77], [80] work on the problem of how to assign questions to only a subset of the sources. [14], [15] pay attention to the sources of informa-

tion, yet focus on the copying relationships between sources. In our setup, no information is copied between users. [53] discusses the problem of how much training data is needed to gain high-quality models. [45] studies truth discovery in quantitative applications, such as percentage annotation and object counting.

Crowdsourcing. The ability discovery problem is closely connected to the truth discovery problem which occur in a wide range of data management problems related to crowdsourcing [34], [79]. Various crowdsourcing systems have been proposed [17], [78], and the crowdsourcing approach has been refined for various tasks, such as query answering [20], entity resolution [8], annotating Twitter data [19], top-k algorithms [75] and various other labeling tasks [25], [63], [67].

Expert finding. The expert finding problem [37], [74] also aims to assess the trustworthiness of users. The difference is that it focuses on *finding experts with expertise (skills) specific to a given question* while our ability discovery problem aims to assess an overall user ability.

VI. CONCLUSIONS

We proposed HITSNDIFFS, a novel variant of HITS, with surprising theoretical and practical properties for ability discovery. On the theoretical side, we showed that 1) C1P of the response matrix models consistent solutions for the problem; 2) our method reconstructs the correct user rankings in the consistent case; 3) does so in linear time and 4) can handle more general cases (in contrast to other linear discrete algorithms). On the practical side, we showed that HnD handles the problem of ability discovery with robust accuracy and greater scalability in terms of the number of users than the only existing C1P reconstruction algorithm that works for general cases.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation (NSF) under award numbers IIS-1762268 and IIS-1956096.

REFERENCES

- [1] “Amazon mechanical turk,” <https://www.mturk.com/>, 2023.
- [2] “Piazza,” <https://piazza.com/>, 2023.
- [3] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigenvalue problem,” *Quarterly of applied mathematics*, vol. 9, no. 1, pp. 17–29, 1951. [Online]. Available: <http://www.jstor.org/stable/43633863>
- [4] J. E. Atkins, E. G. Boman, and B. Hendrickson, “A spectral algorithm for seriation and the consecutive ones problem,” *SIAM Journal on Computing*, vol. 28, no. 1, pp. 297–310, 1998. [Online]. Available: <https://doi.org/10.1137/S0097539795285771>
- [5] A. Birnbaum, “Some latent trait models,” in *Statistical theories of mental test scores*, F. M. Lord and M. R. Novick, Eds. Addison-Wesley, 1968, ch. 17. [Online]. Available: <https://psycnet.apa.org/record/1968-35040-000>
- [6] R. D. Bock, “Estimating item parameters and latent ability when responses are scored in two or more nominal categories,” *Psychometrika*, vol. 37, no. 1, pp. 29–51, 1972. [Online]. Available: <https://doi.org/10.1007/BF02291411>
- [7] K. S. Booth and G. S. Lueker, “Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms,” *Journal of Computer and System Sciences*, vol. 13, no. 3, pp. 335–379, 1976. [Online]. Available: [https://doi.org/10.1016/S0022-0000\(76\)80045-1](https://doi.org/10.1016/S0022-0000(76)80045-1)
- [8] C. Chai, G. Li, J. Li, D. Deng, and J. Feng, “Cost-effective crowdsourced entity resolution: A partial-order approach,” in *SIGMOD*, 2016, pp. 969–984. [Online]. Available: <https://doi.org/10.1145/2882903.2915252>
- [9] Z. Chen, S. Mitra, R. Ravi, and W. Gatterbauer, “HITSnDiffs: From truth discovery to ability discovery by recovering matrices with the consecutive ones property: Code and experiments,” 2023. [Online]. Available: <https://github.com/northeastern-datalab/HITSnDiffs/>
- [10] J. K. Cullum and R. A. Willoughby, *Lanczos algorithms for large symmetric eigenvalue computations: Vol. I: Theory*. SIAM, 2002. [Online]. Available: <https://doi.org/10.1137/1.9780898719192>
- [11] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi, “Aggregating crowdsourced binary ratings,” in *WWW*, 2013, pp. 285–294. [Online]. Available: <https://doi.org/10.1145/2488388.2488414>
- [12] A. P. Dawid and A. M. Skene, “Maximum likelihood estimation of observer error-rates using the EM algorithm,” *Applied statistics*, pp. 20–28, 1979. [Online]. Available: <https://doi.org/10.2307/2346806>
- [13] C. DeMars, *Item Response Theory*. Oxford University Press, 04 2010. [Online]. Available: <https://doi.org/10.1093/acprof:oso/9780195377033.001.0001>
- [14] X. L. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava, “Global detection of complex copying relationships between sources,” *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1358–1369, 2010. [Online]. Available: <https://doi.org/10.14778/1920841.1921008>
- [15] X. L. Dong, L. Berti-Equille, and D. Srivastava, “Integrating conflicting data: the role of source dependence,” *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 550–561, 2009. [Online]. Available: <https://doi.org/10.14778/1687627.1687690>
- [16] X. L. Dong, B. Saha, and D. Srivastava, “Less is more: Selecting sources wisely for integration,” *Proceedings of the VLDB Endowment*, vol. 6, no. 2, pp. 37–48, 2012. [Online]. Available: <https://doi.org/10.14778/2535568.2448938>
- [17] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng, “icrowd: An adaptive crowdsourcing framework,” in *SIGMOD*, 2015, pp. 1015–1030. [Online]. Available: <https://doi.org/10.1145/2723372.2750550>
- [18] M. Fiedler, “Laplacian of graphs and algebraic connectivity,” *Banach Center Publications*, vol. 25, no. 1, pp. 57–70, 1989. [Online]. Available: <https://doi.org/10.4064/-25-1-57-70>
- [19] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze, “Annotating named entities in twitter data with crowdsourcing,” in *NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk*, 2010, pp. 80–88. [Online]. Available: <https://aclanthology.org/W10-0713/>
- [20] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, “CrowdDB: answering queries with crowdsourcing,” in *SIGMOD*, 2011, pp. 61–72. [Online]. Available: <https://doi.org/10.1145/1989323.1989331>
- [21] G. Frobenius, “Über matrizen aus nicht negativen elementen,” *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, pp. 456–477, 1912. [Online]. Available: https://commons.wikimedia.org/wiki/File:Ueber_Matrizen_aus_nicht_negativen_Elementen.pdf
- [22] D. R. Fulkerson and O. A. Gross, “Incidence matrices and interval graphs.” *Pacific Journal of Mathematics*, vol. 15, no. 3, pp. 835 – 855, 1965. [Online]. Available: <http://dx.doi.org/10.2140/pjm.1965.15.835>
- [23] A. Ghosh, S. Kale, and P. McAfee, “Who moderates the moderators?: crowdsourcing abuse detection in user-generated content,” in *EC*, 2011, pp. 167–176. [Online]. Available: <https://doi.org/10.1145/1993574.1993599>
- [24] J. Hauschild and F. Pollmann, “Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy),” *SciPost Phys. Lect. Notes*, p. 5, 2018, code available from <https://github.com/tenpy/tenpy>. [Online]. Available: <https://scipost.org/10.21468/SciPostPhysLectNotes.5>
- [25] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, “Crowdsourced POI labelling: Location-aware result inference and task assignment,” in *ICDE*, 2016, pp. 61–72. [Online]. Available: <https://doi.org/10.1109/ICDE.2016.7498229>
- [26] Y. Hu, J. Scott, and H. MC73, “A fast multilevel fiedler and profile reduction code,” in *RAL-TR-2003-36*, 2003.
- [27] N. Q. V. Hung, H. H. Viet, T. T. Nguyen, M. Weidlich, H. Yin, and X. Zhou, “Computing crowd consensus with partial agreement,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 1, pp. 1–14, 2018. [Online]. Available: <https://doi.org/10.1109/TKDE.2017.2750683>
- [28] H. Kajino, Y. Tsuabei, and H. Kashima, “A convex formulation for learning from crowds,” in *AAAI*, 2012, pp. 73–79. [Online]. Available: <https://doi.org/10.1609/aaai.v26i1.8105>
- [29] D. Kendall, “Incidence matrices, interval graphs and seriation in archeology,” *Pacific Journal of mathematics*, vol. 28, no. 3, pp. 565–570, 1969. [Online]. Available: <https://msp.org/pjm/1969/28-3/p08.xhtml>
- [30] N. M. Kingston and N. J. Dorans, “The feasibility of using item response theory as a psychometric model for the gre aptitude test,” *ETS Research Report Series*, vol. 1982, no. 1, 1982. [Online]. Available: <http://dx.doi.org/10.1002/j.2333-8504.1982.tb01298.x>
- [31] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999. [Online]. Available: <https://doi.org/10.1145/324133.324140>
- [32] C. Lanczos, “An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,” 1950. [Online]. Available: <https://doi.org/DOI:10.6028/JRES.045.026>
- [33] D. Lay, S. Lay, and J. McDonald, *Linear Algebra and Its Applications*. Pearson, 2016. [Online]. Available: <https://books.google.com/books?id=L8SUoAEACAAJ>
- [34] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, “Crowdsourced data management: A survey,” in *ICDE*, 2017, pp. 39–40. [Online]. Available: <https://doi.org/10.1109/ICDE.2017.26>
- [35] J. Li, Y. Baba, and H. Kashima, “Hyper questions: Unsupervised targeting of a few experts in crowdsourcing,” in *CIKM*, 2017, pp. 1069–1078. [Online]. Available: <https://doi.org/10.1145/3132847.3132971>
- [36] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, “A confidence-aware approach for truth discovery on long-tail data,” *Proceedings of the VLDB Endowment*, vol. 8, no. 4, pp. 425–436, 2014. [Online]. Available: <https://doi.org/10.14778/2735496.2735505>
- [37] S. Lin, W. Hong, D. Wang, and T. Li, “A survey on expert finding techniques,” *J. Intell. Inf. Syst.*, vol. 49, no. 2, pp. 255–279, 2017. [Online]. Available: <https://doi.org/10.1007/s10844-016-0440-5>
- [38] Q. Liu, J. Peng, and A. T. Ihler, “Variational inference for crowdsourcing,” in *NIPS*, 2012, pp. 701–709. [Online]. Available: <https://dl.acm.org/doi/10.5555/2999134.2999212>
- [39] F. M. Lord, M. R. Novick, and A. Birnbaum, *Statistical Theories of Mental Test Scores*. Addison-Wesley, 1968. [Online]. Available: <https://psycnet.apa.org/record/1968-35040-000>
- [40] S. Lyu, W. Ouyang, H. Shen, and X. Cheng, “Truth discovery by claim and source embedding,” in *CIKM*. ACM, 2017, pp. 2183–2186. [Online]. Available: <https://doi.org/10.1145/3132847.3133069>
- [41] L. W. Mackey, “Deflation methods for sparse PCA,” in *NIPS*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008, pp. 1017–1024. [Online]. Available: <https://proceedings.neurips.cc/paper/2008/hash/85d8ce590ad8981ca2c8286f79f59954-Abstract.html>
- [42] C. D. Meyer, Ed., *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, 2000. [Online]. Available: <https://doi.org/10.1137/1.9781611977448>
- [43] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022, p. 255. [Online]. Available: probml.ai

- [44] M. Newman, *Networks: An Introduction*. New York, NY, USA: Oxford University Press, Inc., 2010. [Online]. Available: <https://doi.org/10.1093/acprof:oso/9780199206650.001.0001>
- [45] R. W. Ouyang, L. M. Kaplan, A. Toniolo, M. Srivastava, and T. J. Norman, "Aggregating crowdsourced quantitative claims: Additive and multiplicative models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1621–1634, 2016. [Online]. Available: <https://doi.org/10.1109/TKDE.2016.2535383>
- [46] B. N. Parlett and D. S. Scott, "The lanczos algorithm with selective orthogonalization," *Mathematics of Computation*, vol. 33, no. 145, pp. 217–238, 1979. [Online]. Available: <https://doi.org/10.2307/2006037>
- [47] J. Pasterneck and D. Roth, "Knowing what to believe (when you already know something)," in *COLING*, C. Huang and D. Jurafsky, Eds., 2010, pp. 877–885. [Online]. Available: <https://aclanthology.org/C10-1099/>
- [48] O. Perron, "Zur Theorie der Matrices," *Mathematische Annalen*, vol. 64, no. 2, pp. 248–263, 1907. [Online]. Available: <https://doi.org/10.1007/BF01449896>
- [49] M.-T. Puth, M. Neuhaeuser, and G. D. Ruxton, "Effective use of spearman's and kendall's correlation coefficients for association between two measured traits," *Anim. Behav.*, vol. 102, pp. 77–84, April 2015. [Online]. Available: <https://doi.org/10.1016/j.anbehav.2015.01.010>
- [50] G. Rasch, *Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests*. Danmarks Paedagogiske Institut, 1960. [Online]. Available: <https://psycnet.apa.org/record/1962-07791-000>
- [51] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, 2010. [Online]. Available: <https://dl.acm.org/doi/10.5555/1756006.1859894>
- [52] T. Rekatsinas, X. L. Dong, and D. Srivastava, "Characterizing and selecting fresh data sources," in *SIGMOD*, 2014, pp. 919–930. [Online]. Available: <https://doi.org/10.1145/2588555.2610504>
- [53] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. G. Parameswaran, and C. Ré, "Slimfast: Guaranteed results for data fusion and source reliability," in *SIGMOD*, 2017, pp. 1399–1414. [Online]. Available: <https://doi.org/10.1145/3035918.3035951>
- [54] F. Samejima, "Estimation of latent ability using a response pattern of graded scores." *Psychometrika monograph supplement*, 1969. [Online]. Available: <https://doi.org/10.1007/BF003372160>
- [55] ———, "A general model for free-response data." *Psychometrika Monograph Supplement*, 1972. [Online]. Available: <https://psycnet.apa.org/record/1972-28083-001>
- [56] ———, "A new family of models for the multiple-choice item." Tennessee Univ Knoxville Dept of Psychology, Tech. Rep., 1979. [Online]. Available: <https://apps.dtic.mil/sti/citations/ADA080350>
- [57] ———, "Graded response model," in *Handbook of modern item response theory*. Springer, 1997, pp. 85–100. [Online]. Available: https://doi.org/10.1007/978-1-4757-2691-6_5
- [58] R. Sanchez, "Girth: Item Response Theory in Python." [Online]. Available: <https://github.com/eribean/girth>
- [59] A. Sheshadri and M. Lease, "Square: A benchmark for research on computing crowd consensus," in *First AAAI conference on human computation and crowdsourcing*, 2013. [Online]. Available: <https://ojs.aaai.org/index.php/HCOMP/article/view/13088>
- [60] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904. [Online]. Available: <https://doi.org/10.2307/1422689>
- [61] G. W. Stewart and J. guang Sun, *Matrix perturbation theory*. Academic Press, 1990. [Online]. Available: <https://www.worldcat.org/title/matrix-perturbation-theory/oclc/908946968>
- [62] T. Sunahase, Y. Baba, and H. Kashima, "Pairwise HITS: Quality estimation from pairwise comparisons in creator-evaluator crowdsourcing process," in *AAAI*, vol. 31, no. 1, 2017, pp. 977–984. [Online]. Available: <https://doi.org/10.1609/aaai.v31i1.10634>
- [63] A. Tarasov, S. J. Delany, and C. Cullen, "Using crowdsourcing for labelling emotional speech assets," *W3C EmotionML Workshop*, 2010. [Online]. Available: <https://www.w3.org/2010/10/emotionml/papers/tarasov.pdf>
- [64] W. J. Van Der Linden and R. K. Hambleton, "Item response theory: Brief history, common models, and extensions," in *Handbook of modern item response theory*. Springer, 1997, pp. 1–28. [Online]. Available: https://doi.org/10.1007/978-1-4757-2691-6_1
- [65] C. Vania, P. M. Htut, W. Huang, D. A. Mungra, R. Y. Pang, J. Phang, H. Liu, K. Cho, and S. R. Bowman, "Comparing test sets with item response theory," in *ACL/IJCNLP 2021*. Association for Computational Linguistics, 2021, pp. 1141–1158. [Online]. Available: <https://doi.org/10.18653/v1/2021.acl-long.92>
- [66] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. [Online]. Available: <https://doi.org/10.1038/s41592-019-0686-2>
- [67] P. Welinder and P. Perona, "Online crowdsourcing: rating annotators and obtaining cost-effective labels," in *CVPRW*, 2010, pp. 25–32. [Online]. Available: <https://doi.org/10.1109/CVPRW.2010.5543189>
- [68] P. A. White, "The computation of eigenvalues and eigenvectors of a matrix," *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 4, pp. 393–437, 1958. [Online]. Available: <http://www.jstor.org/stable/2098714>
- [69] ———, "Hotelling's matrix deflation," *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 4, pp. 414–415, 1958. [Online]. Available: <http://www.jstor.org/stable/2098714>
- [70] ———, "Wilkinson's vector annihilation," *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 4, p. 414, 1958. [Online]. Available: <http://www.jstor.org/stable/2098714>
- [71] J. Whitehill, T. fan Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvo, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *NIPS*, 2009, pp. 2035–2043. [Online]. Available: <https://dl.acm.org/doi/10.5555/2984093.2984321>
- [72] J. Yang, J. Fan, Z. Wei, G. Li, T. Liu, and X. Du, "A game-based framework for crowdsourced data labeling," *The VLDB Journal*, vol. 29, no. 6, pp. 1311–1336, 2020. [Online]. Available: <https://doi.org/10.1007/s00778-020-00613-w>
- [73] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 796–808, 2008. [Online]. Available: <https://doi.org/10.1109/TKDE.2007.190745>
- [74] S. Yuan, Y. Zhang, J. Tang, W. Hall, and J. B. Cabotà, "Expert finding in community question answering: a review," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 843–874, 2020. [Online]. Available: <https://doi.org/10.1007/s10462-018-09680-6>
- [75] X. Zhang, G. Li, and J. Feng, "Crowdsourced top-k algorithms: An experimental evaluation," *PVLDB*, vol. 9, no. 8, pp. 612–623, 2016. [Online]. Available: <http://www.vldb.org/pvldb/vol9/p612-zhang.pdf>
- [76] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan, "Spectral methods meet EM: A provably optimal algorithm for crowdsourcing," *J. Mach. Learn. Res.*, vol. 17, pp. 102:1–102:44, 2016. [Online]. Available: <http://jmlr.org/papers/v17/14-511.html>
- [77] Y. Zheng, R. Cheng, S. Maniu, and L. Mo, "On optimality of jury selection in crowdsourcing," in *EDBT*. OpenProceedings.org, 2015, pp. 193–204. [Online]. Available: <https://doi.org/10.5441/002/edbt.2015.18>
- [78] Y. Zheng, G. Li, and R. Cheng, "DOCS: domain-aware crowdsourcing system," *PVLDB*, vol. 10, no. 4, pp. 361–372, 2016. [Online]. Available: <http://www.vldb.org/pvldb/vol10/p361-zheng.pdf>
- [79] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: Is the problem solved?" *PVLDB*, vol. 10, no. 5, pp. 541–552, 2017. [Online]. Available: <https://doi.org/10.14778/3055540.3055547>
- [80] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "QASCA: A quality-aware task assignment system for crowdsourcing applications," in *SIGMOD*, 2015, pp. 1031–1046. [Online]. Available: <https://doi.org/10.1145/2723372.2749430>

APPENDIX A NOMENCLATURE

m	number of users
n	number of items
k	number of choices (or options) for each item
u_j	user j
θ_j	ability of user j
t_i	item i
c_{ih}	choice (or option) h for item i
a_{ih}	discrimination of option h for item i in discrimination-difficulty parameterization
b_{ih}	difficulty of option h for item i in discrimination-difficulty parameterization
α_{ih}	discrimination (or slope) of option h for item i in slope-intercept parameterization
β_{ih}	difficulty (or intercept) of option h for item i in slope-intercept parameterization
c_i	random guessing parameter in some IRT models for item i
$\mathbf{A}_{:j}$	j^{th} row of some matrix \mathbf{A}
$\mathbf{A}_{:i}$	i^{th} column of some matrix \mathbf{A}
A_{ji}	(j, i) entry of matrix \mathbf{A}
λ_i	the i -th largest eigenvalue
\mathbf{v}_i	the eigenvector corresponding to the i -th largest eigenvalue (for simplicity, also referred to as “ i -th largest eigenvector”)
\mathbf{C}	$(m \times kn)$ binary response matrix
\mathbf{D}	the diagonal matrix of a square matrix. Of size $(m \times m)$ when it represents the diagonal matrix of \mathbf{CC}^\top used by ABH [4]
\mathbf{L}	the Laplacian matrix of a square matrix. Of size $(m \times m)$ when it represents the Laplacian matrix of \mathbf{CC}^\top used by ABH [4]: $\mathbf{L} = \mathbf{D} - \mathbf{CC}^\top$
\mathbf{s}	(m) user score vector where s_j denotes score of user j
\mathbf{w}	(kn) option weight vector where w_i denotes the weight of option $((i-1) \bmod k) + 1$ for item $((i-1) \bmod k) + 1$
\mathbf{e}	the ones vector $\mathbf{1}_r$ for some constant $r \in \mathbb{N}^+$
\mathbf{C}^{col}	$(m \times kn)$ column-normalized response matrix
\mathbf{C}^{row}	$(m \times kn)$ row-normalized response matrix
\mathbf{U}	$(m \times m)$ update matrix $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^\top$ used by AVGHITS
\mathbf{U}^{diff}	$((m-1) \times (m-1))$ difference update matrix $\mathbf{U}^{\text{diff}} = \mathbf{SUT}$ used by HND to update user score differences \mathbf{s}^{diff}
\mathbf{s}^{diff}	$(m-1)$ user difference vector with $s_j^{\text{diff}} = s_{j+1} - s_j, j \in [m-1]$
\mathbf{S}	$((m-1) \times m)$ matrix to compute the differences in a vector
\mathbf{T}	$(m \times (m-1))$ matrix to reconstruct a vector based on the differences
\mathbf{M}	$((m-1) \times (m-1))$ matrix $\mathbf{M} = \mathbf{SLT}$
\mathbf{I}	identity matrix (quadratic matrix with 1's on the diagonal)
t	number of iterations
\mathbf{e}_i	the vector with the i^{th} element 1 and all other elements 0
$[m]$	set $\{1, 2, \dots, m\}$

APPENDIX B PROOFS

In this section, we prove our main [Theorem 2](#). Before the proof, we first review some background.¹⁵

The method of Atkins et al. (ABH). Given a pre-P matrix \mathbf{C} with a unique consecutive ones ordering, the spectral method of Atkins et al. [4] (which we refer to as ABH) finds this ordering of users that reconstruct the C1P property for \mathbf{C} by using the product of the response matrix \mathbf{C} and its transpose

¹⁵The references in the appendix refer to the full reference, which includes 5 more papers.

to form \mathbf{CC}^\top . The method has two main parts: showing that if \mathbf{C} is a P-matrix, then (1) \mathbf{CC}^\top has the special property of being an R-matrix (which we will define below), and (2) the 2nd smallest eigenvector of the related Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{CC}^\top$ is monotonic (i.e. its elements are in either increasing or decreasing order).

Here \mathbf{D} is a diagonal matrix filled with the row sums of \mathbf{CC}^\top . Given these two facts, they prove that if \mathbf{C} is a pre-P matrix, then the row permutations induced by sorting the values of 2nd smallest eigenvector of \mathbf{L} can reconstruct the C1P property for \mathbf{C} .

Definition 4 (R-matrix [4]). *A matrix \mathbf{A} is called an R-matrix if it is symmetric and*

$$\begin{aligned} A_{ji} &\geq A_{jh} & \text{for } j < i < h \\ A_{ji} &\leq A_{jh} & \text{for } i < h < j \end{aligned}$$

Intuitively, the definition describes a matrix where the values fall off as we move away from the diagonal along any row. Since for \mathbf{CC}^\top , each matrix entry represents the dot product of the responses of two users. The entries represent the number of common responses for a pair of users. When the response matrix is sorted by user ability, these values are highest for users with themselves and fall off as we move away along the correct linear ordering of the abilities represented in the correctly sorted response matrix.

Our method. Given a pre-P matrix \mathbf{C} with a unique consecutive ones ordering, we use our update matrix $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^\top$ to find this orderings of users that reconstruct the C1P property for the matrix \mathbf{C} . To do this, we will prove the following two statements: (1) [Lemma 6](#) shows that if \mathbf{C} is a P-matrix, then \mathbf{U} is an R-matrix. (2) [Lemma 7](#) shows that the 2nd largest eigenvector of \mathbf{U} is monotonic. These two statements imply that if \mathbf{C} is a pre-P matrix, then the eigenvector corresponding to the 2nd largest eigenvalue of \mathbf{U} can be used to find the unique ordering that reconstruct the C1P property for the matrix \mathbf{C} .

Without loss of generality, we assume that every item option was chosen by at least one student. If an option was never chosen, this option has no information and can be removed. Thus in the following proof, we assume that every column of \mathbf{C} contains at least one 1.

We first recall a well-known lemma that we will use in the proofs.

Lemma 2 (Constant row sums). *If all rows of a non-negative square matrix \mathbf{A} sum to a scalar b , then the largest eigenvalue of \mathbf{A} is b with the corresponding eigenvector in the direction of $\mathbf{e} = \mathbf{1}_n$.*

Proof of Lemma 2. It is easy to see that \mathbf{e} is an eigenvector with eigenvalue b : since every row in \mathbf{A} sums to b , every entry in $\mathbf{A}\mathbf{e}$ is b . That b is the largest eigenvalue follows from the fact that the spectral radius $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$ for every matrix norm [42], including the induced ∞ -norm $\|\mathbf{A}\|_\infty$ which is the maximum absolute row sum. \square

Note that if there are multiple connected components in the user-option bipartite graph, there is no way to get a total ordering on the users or items, since we cannot compare between the different connected components; we can only get an ordering for each connected component *separately*. Thus, in the sequence, we assume a single connected component, and thus, multiplicity 1 of the largest eigenvalue of the update matrix \mathbf{U} .

Lemma 3 (\mathbf{U} is row-stochastic). *Each row of \mathbf{U} has sum 1.*

Proof of Lemma 3. Define r_j as the sum of elements of the j^{th} row of $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}$, i.e. $r_j = \sum_{i=1}^m U_{ji}$. We want to show that $r_j = 1, \forall j \in [m]$.

Since $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\top}$, the j^{th} row of \mathbf{U} , $\mathbf{U}_{j:}$, is created from the corresponding row $\mathbf{C}_{j:}^{\text{row}}$ of \mathbf{C}^{row} and all the rows $\mathbf{C}_{i:}^{\text{col}}$ of \mathbf{C}^{col} :

$$r_j = \sum_{i=1}^m \mathbf{C}_{j:}^{\text{row}} \cdot ((\mathbf{C}^{\text{col}})^{\top})_{:i} = \sum_{i=1}^m \mathbf{C}_{j:}^{\text{row}} \cdot \mathbf{C}_{i:}^{\text{col}} = \mathbf{C}_{j:}^{\text{row}} \cdot \sum_{h=1}^m \mathbf{C}_{i:h}^{\text{col}}$$

But recall that by construction, $\sum_{i=1}^m \mathbf{C}_{i:}^{\text{col}}$ is the ones vector \mathbf{e} (every option is chosen by at least one student or otherwise the whole column is removed). So,

$$r_j = \mathbf{C}_{j:}^{\text{row}} \cdot \sum_{i=1}^m \mathbf{C}_{i:}^{\text{col}} = \mathbf{C}_{j:}^{\text{row}} \cdot \mathbf{e} = 1$$

The last step follows from the definition of the row-normalized binary response matrix \mathbf{C}^{row} . \square

Lemma 4 (1st eigenvector of AVGHITS). *If the largest eigenvalue of \mathbf{U} has multiplicity 1, the fixed point of the AVGHITS update rule is in the direction of $\mathbf{e} = \mathbf{1}_m$*

Proof of Lemma 4. The AVGHITS update rules above imply for any iteration $t > 0$,

$$\begin{aligned} \mathbf{s}^{(t)} &= \mathbf{U}\mathbf{s}^{(t-1)} \\ \implies \mathbf{s}^{(t)} &= \mathbf{U}^t \mathbf{s}^{(0)} \end{aligned}$$

Let λ_1 be the largest eigenvalue of \mathbf{U} with the corresponding eigenvector \mathbf{v}_1 . By the power rule, if λ_1 has multiplicity 1, $\mathbf{s}^{(t)}$ converges in the direction of \mathbf{v}_1 as t goes to infinity (barring the very unlikely random initialization of $\mathbf{s}^{(0)}$ that is orthogonal to \mathbf{v}_1).

Note that \mathbf{U} is square and non-negative since \mathbf{C}^{row} and \mathbf{C}^{col} are both non-negative. By Lemma 3, each row of \mathbf{U} has sum 1, and we can then use Lemma 2. This gives us the result, since the fixed point of AVGHITS is in the direction of the eigenvector corresponding to the largest eigenvalue λ_1 of \mathbf{U} (provided λ_1 has multiplicity 1) and Lemma 2 implies \mathbf{U} has largest eigenvalue 1 with the corresponding eigenvector \mathbf{e} . \square

We can now prove our main result in Theorem 1 using the following three lemmas. In those lemmas, we make the simplifying assumption that each row of the matrix \mathbf{C} has the same row sum (i.e. the same number of 1s in each row), which means each user chooses the same number of options. This is WLOG, because given any arbitrary matrix \mathbf{C} , we can

add additional columns with exactly one 1 and $m - 1$ zeros each to the matrix \mathbf{C} until each row has the same number of 1s. Note that adding such columns does not affect the C1P property, since each additional column has only a single 1. In practice, the ordering of users might be affected by padding the matrix with columns in this way, so we only make this assumption of equal row sums for proving results related to the C1P property of the ideal case.

Lemma 5. *If the response matrix \mathbf{C} is a P-matrix and each row has the same row sum, the update matrix \mathbf{U} is symmetric.*

Proof of Lemma 5. First we assume each user chooses n^{ans} options. Consider any $i, j \in [m], i \neq j$. We need to show that $U_{ij} = U_{ji}$, where

$$\begin{aligned} U_{ji} &= \sum_{h=1}^{\text{width}(\mathbf{C})} C_{jh}^{\text{row}} C_{ih}^{\text{col}} \\ U_{ij} &= \sum_{h=1}^{\text{width}(\mathbf{C})} C_{ih}^{\text{row}} C_{jh}^{\text{col}} \end{aligned}$$

To contribute to the sum, the respective elements of \mathbf{C}^{row} and \mathbf{C}^{col} have to both be nonzero. For all $i \neq j$, C_{jh}^{row} and C_{ih}^{col} being both nonzero implies C_{jh} and C_{ih} are 1, which also means C_{ih}^{row} and C_{jh}^{col} are both nonzero.

Because each user answers the same number of questions n^{ans} , all nonzero entries in \mathbf{C}^{row} are equal, and therefore also $C_{jh}^{\text{row}} = C_{ih}^{\text{row}} = \frac{1}{n^{\text{ans}}}$ if both are nonzero. Also by definition of \mathbf{C}^{col} , $C_{jh}^{\text{col}} = C_{ih}^{\text{col}}$ if they are both nonzero since they appear in the same column of \mathbf{C}^{col} . Combining the two equations, $C_{jh}^{\text{row}} C_{ih}^{\text{col}} = C_{ih}^{\text{row}} C_{jh}^{\text{col}}$ for all $i \neq j$. Therefore, also $U_{ij} = U_{ji}$ for all $i \neq j$, and thus \mathbf{U} is symmetric. \square

Lemma 6. *If the response matrix \mathbf{C} is a P-matrix and each row has the same row sum, then the update matrix \mathbf{U} is an R-matrix.*

Proof of Lemma 6. We need to show that neither of

$$U_{ji} > U_{jk} \quad \text{for } i < k < j \tag{4}$$

$$U_{ji} < U_{jk} \quad \text{for } j < i < k \tag{5}$$

is true, where

$$\begin{aligned} U_{ji} &= \sum_{h=1}^{\text{width}(\mathbf{C})} C_{jh}^{\text{row}} C_{ih}^{\text{col}} \\ U_{jk} &= \sum_{h=1}^{\text{width}(\mathbf{C})} C_{jh}^{\text{row}} C_{kh}^{\text{col}} \end{aligned}$$

Analogous to the proof for the symmetry (Lemma 5), we have $C_{jh}^{\text{row}} C_{ih}^{\text{col}} = C_{jh}^{\text{row}} C_{kh}^{\text{col}}$ for all nonzero C_{jh}, C_{ih}, C_{kh} . In order for (4) to hold, there would have to be more non-zero $C_{jh}^{\text{row}} C_{ih}^{\text{col}}$ than $C_{jh}^{\text{row}} C_{kh}^{\text{col}}$. This in turn would imply that among the columns of \mathbf{C} which have a 1 in the j^{th} row (and therefore nonzero entries in the j^{th} row of \mathbf{C}^{row}) there are strictly more 1's in the i^{th} row than the k^{th} row. But since $i < k < j$, the

column which has a 1 in the i th row but not the k th row leads to a violation of the consecutive ones property.

(5) leads to a similar contradiction. \square

Lemma 7. *If C is a P-matrix and each row has the same row sum, the 2nd largest eigenvector of U is monotonic.*

Proof of Lemma 7. We will prove this in a way similar to [4]. Define the matrices $S \in \mathbb{R}^{(m-1) \times m}$ and $T \in \mathbb{R}^{m \times (m-1)}$ as in [Section III-C](#).

Note that $TS = (\mathbf{I}_m - \mathbf{e}\mathbf{e}_1^\top)$. Note that for any vector v , $Sv = (v_2 - v_1, v_3 - v_2, \dots, v_r - v_{r-1})^\top$. Similarly, the i th row of SU is just the difference between the $(i+1)$ th and i th rows of U . However, as shown in [Lemma 3](#), each row of U sums to 1. This implies each row of SU sums to 0.

Let x be an eigenvector of U that is not in the direction of the all ones vector $e = \mathbf{1}_m$, i.e. $x \neq \alpha e$. Then,

$$\begin{aligned} Ux &= \lambda x \\ SUx &= \lambda Sx \\ SU(\mathbf{I}_m - \mathbf{e}\mathbf{e}_1^\top)x &= \lambda Sx \\ SUTSx &= \lambda Sx \\ U^{\text{diff}}y &= \lambda y, \quad \text{where } y = Sx \end{aligned} \tag{6}$$

The equivalence between the 2nd and 3rd lines above is because each row of SU sums to 0 as shown above, so $SUe = \mathbf{0}$. So, any eigenvalue λ of U is also an eigenvalue of U^{diff} , except for the eigenvalue of U corresponding to the e eigenvector leading to $Se = 0$. Therefore, U^{diff} has exactly the same eigenvalues as U except the largest eigenvalue 1, and the eigenvectors of U^{diff} are the differences between the entries of the corresponding eigenvector of U , which proves [Lemma 1](#).

Then we prove that U^{diff} is a non-negative matrix. First, we think about SU . As we explained before, every row of it sums to 0. Moreover, for each the r th row of SU which is the difference between the r th and $(r+1)$ th row of U , the first r entries are non-positive while all the others are non-negative because U is an R-matrix.

Then we turn to the multiplication between SU and T , which generates U^{diff} . The entry at the j -th row and the i -th column of U^{diff} is the product of the j -th row of SU and the i -th column of T . This product is actually the sum of the last $(m-i)$ entries of the j -th row of SU , which also equals to the sum of the j -th row of SU minus the first i entries. Recall that the first r entries are non-positive while all the others are non-negative for the r -th row of SU , and each row of SU sums to 0. If $j \geq i$, the first i entries of the j -th row of SU are all non-positive so the product is non-negative. If $j < i$, the last $m-i$ entries of the j -th row of SU are all non-negative so the product is non-negative.

Therefore, we prove every entry in U^{diff} is non-negative, which means U^{diff} is a non-negative matrix.

We can now apply the Perron-Frobenius Theorem [21], [48]: there exists a non-negative eigenvector of U^{diff} corresponding to the largest eigenvalue of U^{diff} . Note that U^{diff} has exactly the same eigenvalues as U , except the largest

eigenvalue 1, and the eigenvectors of U^{diff} are the differences between the elements of the corresponding eigenvector of U . Since the differences between the elements of the eigenvector corresponding to the 2nd largest eigenvalue of U (largest eigenvalue of U^{diff}) are non-negative, that eigenvector of U is monotonic. \square

The above three lemmas imply that if we start with a pre-P matrix C with a consistent row sum, sorting the rows according to the second eigenvector ordering of the corresponding update matrix U gives a P-matrix, proving [Theorem 1](#).

From [Lemma 1](#), we know by converting the converged largest eigenvector of U^{diff} back into a user score, we regain the ordering of the rows according to values in the second largest eigenvector of U . This, along with [Theorem 1](#), proves [Theorem 2](#) that an algorithm for computing the largest eigenvector of U^{diff} (HND detailed in [Algorithm 1](#)) reconstructs the ideal consistent ordering.

APPENDIX C ITEM RESPONSE THEORY (IRT) MODELS

[Section C-A](#) starts with the binary (or dichotomous) models that model the probability that a student answers a question correctly. [Section C-B](#) then discusses multinomial (or polytomous) models that model the probability for a student to choose a specific option of a question.

A. Binary (or Dichotomous) Models

The binary IRT models are variations of binary logistic regression with one latent trait x : $\mathbb{P}(x) = \sigma(\alpha x + \beta)$. The intuition is that the probability of a correct answer increases with a student's ability and decreases with the question's difficulty. The *ability* of a student is captured by a single latent factor θ .

1PL (1-Parameter Logistic) model. This is the simplest logistic IRT model and is also called the *Rasch model* [50]. It is called 1PL as it has only one parameter b_i called difficulty for every question i which shifts the logistic function with regard to student abilities. The probability for a user with ability θ to answer a question i correctly is given by the following response function:

$$\mathbb{P}_i(\theta) = \sigma(\theta - b_i) = \frac{1}{1 + e^{-(\theta - b_i)}}$$

Thus the probability of getting a question correct increases with a larger student ability θ and a smaller question difficulty b_i .

2PL (2-Parameter Logistic) model [5]. The 2PL model adds a second parameter a_i called discrimination to every question i . Intuitively, the discrimination models how well a question can separate competent from less competent students: High discrimination implies that the probability of answering it correctly increases strongly with student ability θ . The new response function is:

$$\mathbb{P}_i(\theta) = \sigma(a_i(\theta - b_i)) = \frac{1}{1 + e^{-a_i(\theta - b_i)}}$$

Notice this is exactly the logistic function after changing from *slope-intercept* $\sigma(\alpha_i\theta + \beta_i)$ to *discrimination-difficulty* parameterization $\sigma(a_i(\theta - b_i))$. In other words, the 2PL model is *isomorphic to logistic regression*. It specializes into the 1PL model by tying all discrimination parameters to the same value $a_i = 1$.

GLAD. This model was proposed in the crowdsourcing literature [71] and also uses the logistic function. Interestingly, it is a specialization of the 2PL model with all difficulty parameters tied by $b_i = 0$. A special property of the GLAD model is that for a student whose ability is 0, the probability to answer any question correctly is 50%. The new response function is:

$$\mathbb{P}_i(\theta) = \sigma(a_i\theta) = \frac{1}{1 + e^{-a_i\theta}}$$

3PL (3-Parameter Logistic) model [5]. The 3PL model adds a third parameter c_i to each question i that model the probability that a student without low ability can randomly guess the correct answer. This is motivated in a multiple-choice question (MCQ) setting where the best strategy for a student who does not know the answer is to pick one answer randomly. A reasonable setting for c is $1/k$, where k is the number of choices (or options). The 3PL model specializes to the 2PL model by tying all random guesses to 0: $c_i = 0$. The new response function is

$$\mathbb{P}_i(\theta) = c_i + (1 - c_i)\sigma(a_i(\theta - b_i)) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta - b_i)}}$$

B. Multinomial (Polytomous) Models

Multinomial IRT models model the probability of a student picking a particular choice among k options. They rely on a generalisation of the logistic function to multiple inputs called the softmax activation function (or multinomial logit), used in multinomial logistic regression. The softmax $\sigma : \mathbb{R}^k \rightarrow [0, 1]^k$ defines a probability of a choice among several outcomes for $k \geq 2$ as $\sigma(\mathbf{x})_h = e^{x_h} / \sum_{l=1}^k e^{x_l}$ for $h = 1 \dots k$. The various multinomial IRT models can thus be seen as variants of multinomial logistic regression. We next discuss three important polytomous models and their connections.

Bock. Bock's nominal category model [6] is exactly multinomial logistic regression in slope-intersection parameterization. In this model, each option h has a discrimination parameter α_h and an intercept β_h . The probability for a user with ability θ to pick option h for question i is given by the following response function:

$$\mathbb{P}_{ih}(\theta) = \frac{e^{\alpha_{ih}\theta + \beta_{ih}}}{\sum_{l=0}^{k-1} e^{\alpha_{il}\theta + \beta_{il}}} \quad (7)$$

The option with the largest α is the correct option (students with large enough ability θ will always choose that option). Also, notice that the function is actually over-parameterized: Dividing by $e^{\alpha_{i0}\theta + \beta_{i0}}$ gives a normalized representation with $2(k - 1)$ independent parameters per question. With the same normalization, Bock recovers the binary 2PL model for $k = 2$. (see e.g. [43, section 2.5.3]).

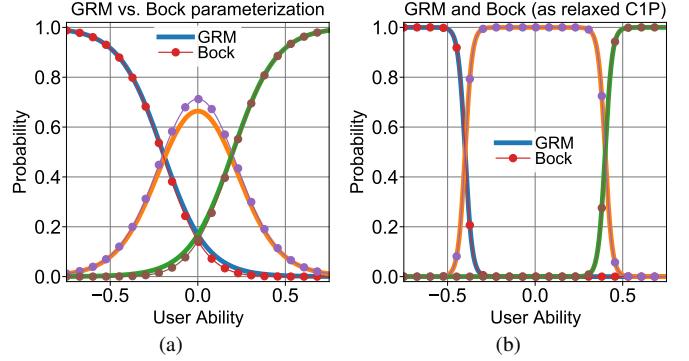


Fig. 8: (a) Example illustrating that GRM, for all practical purposes can be seen as a special case of Bock. Shown are GRM with $a = 8$, $b_h = (-0.2, 0.2)$ and Bock with $\alpha_h = (8, 16)$, $\beta_h = (1.6, 0)$. Three curves correspond to the probability for users to choose three options respectively. (b): Example illustrating that GRM and Bock (like all IRT models discussed in this paper) can be seen as relaxed versions of a response matrix with the C1P property. Shown here are GRM with $a = 50$, $b_h = (-0.4, 0.4)$ and Bock with $\alpha_h = (50, 100)$, $\beta_h = (20, 0)$. Notice the similarity with Figure 1(b) after accounting for the linear ordering of the responses.

Graded Response Model (GRM) [54], [55], [57]. The graded-response model deals with *ordered* polytomous categories such as ratings (e.g. strongly disagree, disagree, agree, and strongly agree, used in attitude surveys). GRM postulates that there are $k + 1$ steps per question $(0, \dots, k)$. A student choosing an option $h \in \{0, \dots, k - 1\}$ passes the first $h + 1$ steps but fails in step $h + 2$. Thus every student passes the first step but no student passes the last step. The model uses the 2PL response function as a cumulative probability function for a student with ability θ to pass step h as:

$$\mathbb{P}_{ih}^*(\theta) = \sigma(a_i(\theta - b_{ih})) = \frac{1}{1 + e^{-a_i(\theta - b_{ih})}}$$

$$-\infty = b_{i0} < b_{i1} < \dots < b_{ik-1} < b_{ik} = \infty$$

where $\mathbb{P}_{ih}^*(\theta) = \sum_{l=h}^k \mathbb{P}_{il}(\theta)$. The option response function is then $\mathbb{P}_{ih} = \mathbb{P}_{ih}^* - \mathbb{P}_{i,h+1}^*$. For a question i with k options, there are k free parameters in all including a discrimination parameter a_i and $k - 1$ difficulty or location parameters $b_{ih}, h \in \{1, \dots, k - 1\}$.

Notice that the logits in the cumulative response function \mathbb{P}^* for GRM all have the same slopes a_i , which is described as the *homogeneous case* of the graded response model [57]. While the correspondence is not exact, GRM can be interpreted as an approximate special case of the Bock model (see Figure 8).

Samejima. Similar to what 3PL adds to 2PL, Samejima [56] introduced a model with *random guessing* based on the Bock model. The model postulates a latent “don't-know” choice numbered zero along with the given k options. A student with low enough ability will guess randomly among the options. Notice that this model does not exactly specialize to the 3PL model for $k = 2$. Unlike 3PL which is able to set a random guessing probability, the Samejima model fixes the probability of a student randomly picking the correct answer to be $1/k$.

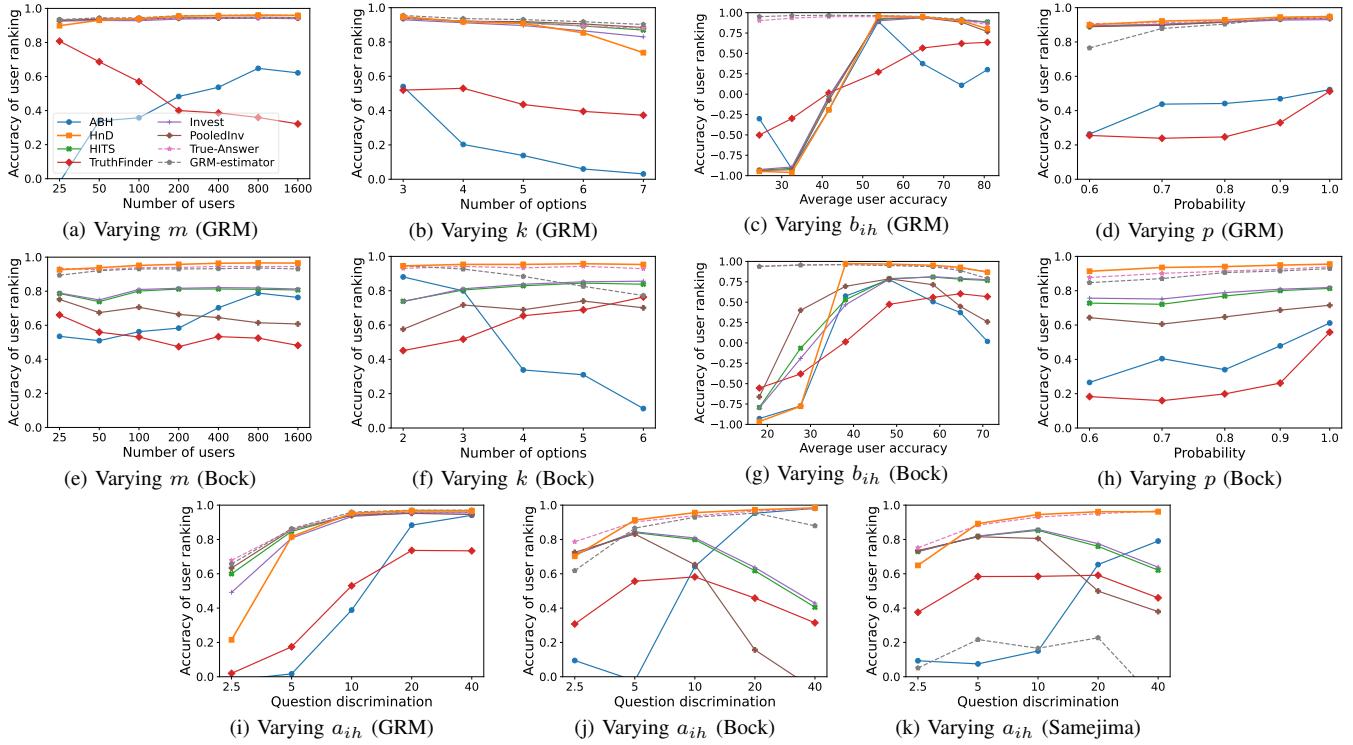


Fig. 9: [Section D-A](#): Results of supplementary accuracy experiments (the legend is in the first figure).

The probability for a user with ability θ to pick an option h for a question i is given by the response function:

$$\mathbb{P}_{ih}(\theta) = \frac{e^{\alpha_{ih}\theta + \beta_{ih}} + \frac{e^{\alpha_{i0}\theta + \beta_{i0}}}{k}}{\sum_{l=0}^k e^{\alpha_{il}\theta + \beta_{il}}}$$

For $\beta_0 \rightarrow -\infty$, this model recovers the Bock model.

APPENDIX D SUPPLEMENTARY EXPERIMENTS

A. Accuracy on synthetic data

In addition to all experiments we show in [Section IV-B](#), we also conduct an experiment to determine the accuracy as function of option discrimination a_{ih} . Moreover, in [Section IV-B](#), we focus on the most general polytomous IRT model, Samejima to avoid redundancy. Here, we also present the results on the other two models GRM and Bock based on the same setup we use in [Section IV-B](#).

Supplementary accuracy experiments on Bock and GRM data ([Figures 9a to 9h](#)). In [Section IV-B](#), [Figures 4a](#) and [4b](#) show the robust performance of HND for datasets generated by Bock and GRM model with varying n but for other experiments we focus on the most general model, Samejima. Here we provide supplementary experimental results on data generated by Bock and GRM model with the same setup as what we do on data generated by the Samejima model in [Section IV-B](#) (corresponding to [Figures 4d to 4g](#)). The experimental results generally fits our observation in [Section IV-B](#).

One major exception appears in [Figures 9c](#) and [9g](#). There we see that when the average accuracy is low for Bock and GRM model, HND gives the approximate reverse ranking. The reason is when the difficulty is larger than the user ability, good users do not return consistent correct answers but worse users all choose the worst answer for such models without random guessing. Thus all the methods tend to believe the majority in such arguably unrealistic setup. Still, we see that for Samejima, HND performs robustly well because Samejima models random guessing so when the difficulty is high, the bad users tend to answer random option so all methods know how to find the good users. In all the realistic scenarios, we see HND outperforms other competitors.

Varying question discriminations a_{ih} ([Figures 9i to 9k](#)). Recall from [Section II-D](#) that larger discriminations imply that a question can more easily “discriminate” between able and less able users, and the correct option can more easily be picked by good users. Here, we change the discrimination range from the default $[0, a_{\max}]$ with $a_{\max} = 10$ to 5 different ranges, $a_{\max} \in \{2.5, 5, 10, 20, 40\}$. We see that HND keeps high accuracy except when $a_{\max} = 2.5$. When a_{\max} is small, the performance between good users and bad users are close, which means the dataset is not good at evaluating users.

B. Accuracy on real-world data

Here we include the details of the real datasets used in [Figure 10](#) and the experimental results on each individual

Dataset	#users	#questions	#options	result
Chinese	50	24	5	Figure 11a
English	63	30	5	Figure 11b
IT	36	25	4	Figure 11c
Medicine	45	36	4	Figure 11d
Pokemon	55	20	6	Figure 11e
Science	111	20	5	Figure 11f

Fig. 10: Summary of real datasets

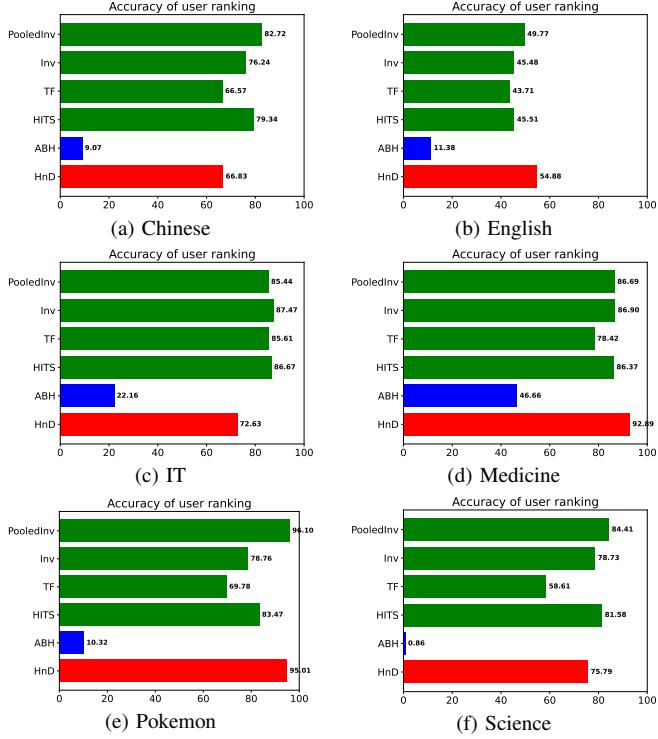


Fig. 11: Detailed result on each dataset in [Figure 7](#)

dataset of [Figure 7](#).¹⁶

We like to remind that these datasets *do not come with a ground truth on user ranking* and we instead inferred an approximate ranking by using the “True answer” baseline explained in [Section IV-A](#) as the reference ranking against which to compare. We have seen in the earlier experimental results that sometimes HND gives better answers than “True answer.” Thus the experimental results should be seen only as a supplementary evidence.

C. Accuracy on realistic simulated data

As mentioned in [Section IV-A](#), we do not know *any existing benchmark with a known true ranking* of users by their abilities. In order to still verify the performance of HND in an as-realistic-as-possible setting, we simulate real-world data by generating synthetic data that follow prior published IRT

¹⁶For the first two experiments, the correlation of ABH is actually negative. We use its absolute value for the simplicity of presentation.

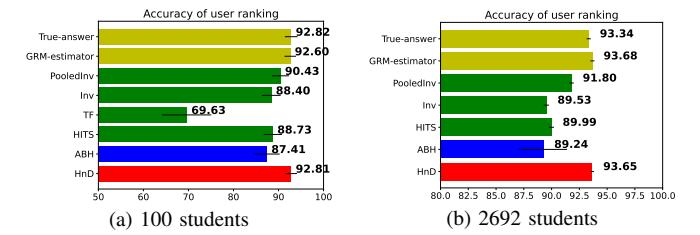


Fig. 12: [Section D-C](#): Accuracy experiments on synthetically simulated data with correlation of user rankings following parameters from real world data described in [13]. Shown are average and standard deviation over 10 runs.

parameters of real-world data [13] or estimated distribution [65]. In this way, we have realistic data with the ground truth.

Experiments on simulated American Experience test data. DeMars [13] presents a detailed analysis on the American Experience test with 40 questions and 2692 participating students. On page 87, the book presents estimates of parameters for a binary 3PL IRT model. We generate 10 small datasets with 100 students and 40 questions, fitting the size of a real-world class and 10 large datasets with 2692 students and 40 questions following the original dataset size provided in [13]. In datasets of both settings, the 40 questions follow exactly those estimated parameters from the book to simulate a realistic scenario, and the student (user) abilities follow the normal distribution $N(0, 1)$ as indicated in [13]. We then conduct experiments on each of them and take the average accuracy. Our results in [Figure 13b](#) clearly show the stronger and more stable performance of HND compared to other competitors except for the cheating estimators.

Experiments on simulated half-moon data. Besides the simulated data based on the data distribution of the American Experience test, which exactly fits our background of students and questions, we also explore the performance of HND on other data. Prior work [65] provides a detailed analysis of IRT parameters over 29 datasets. They consider different natural language understanding models as users in our scenario and use the performances of those models to estimate the binary 3PL IRT parameters ([Section C-A](#)) of the tasks (questions in our scenario) in the datasets. One important take-away from the paper is that the distribution of log discrimination versus difficulty tends to have a half-moon pattern, which means most of the discriminative questions are either easy or difficult. Following this, we develop a simulated question generator with the half-moon distribution of log discrimination versus difficulty as shown in [Figure 13a](#). As for other parameters, we set user ability to follow the normal distribution $N(0, 1)$ and random guessing c to be within $[0, 0.5]$ as hinted by [65]. We generate 10 datasets with 100 users and 100 questions, conduct experiments on them and take the average accuracy as the experimental result. [Figure 13b](#) presents the strong performance of HND, which shows that HND may be also applicable to other crowdsourcing tasks.

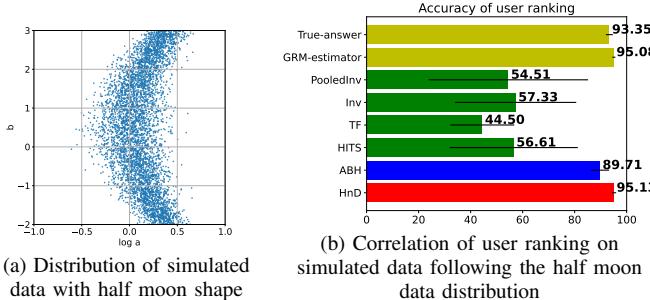


Fig. 13: Experiments on the simulated half moon data

D. Implementation details

To get an approximate correspondence in discrimination between Bock and GRM models, when we parameterize a_{ih} for Bock from $[0, x]$, we also parameterize a_i for GRM from $[0, \frac{2x}{k+1}]$. In this way, the average a_{ih} for Bock is $\frac{x}{2}$ and the average GRM's a_i is $\frac{x}{k+1}$. As discussed in Section II-D, $a_i = \frac{x}{k+1}$ for GRM is approximately the same case as $a_{ih} = \frac{hx}{k+1}$ for Bock for each h . This means $a_i = \frac{x}{k+1}$ for GRM approximates the case that the average a_{ih} for Bock equals $(\sum \frac{hx}{k+1})/k = \frac{(k+1)kx}{2(k+1)}/k = \frac{x}{2}$. In this way, we get similar average discriminations for Bock and GRM.

Another minor detail is that when we generate the C1P data, directly applying the distribution where all user abilities are distributed evenly within $[0, 1]$ will result in completely symmetric response matrix and thus no method (even with symmetry breaking) could possibly determine the side with better users. Therefore we set 10% of the users to be within $[0, 0.5]$ and 90% users to be within $[0.5, 1]$ without influencing the property that the response matrices are consistent.

APPENDIX E DETAILED ANALYSIS

A. Dawid-Skene (DS) for homogenous items

Besides the Item Response Theory, another widely used model for item labeling was proposed by Dawid and Skene (DS) [12]. Here we first introduce the DS model and compare it against the IRT models.

The DS model assumes *homogenous items*, i.e. they are of the same type. It is parameterized by one latent stochastic confusion matrix per user where off-diagonal entries represent the probabilities that a user mislabels an item from one class with another while the diagonal elements correspond to probabilities of making accurate choices. For example, assume $m = 5$ users have to choose one of $k = 3$ labels dog, cat, or rabbit for each of $n = 10$ images. One entry in the confusion matrix for a user specifies the probability that that user chooses the label ‘dog’ when the true label is ‘cat.’ The approach assumes *identical parameters for each item* and thus requires $mk(k-1)$ parameters for m users. The confusion matrices and true labels are jointly estimated by maximizing the likelihood of observed labels via the EM algorithm [12].

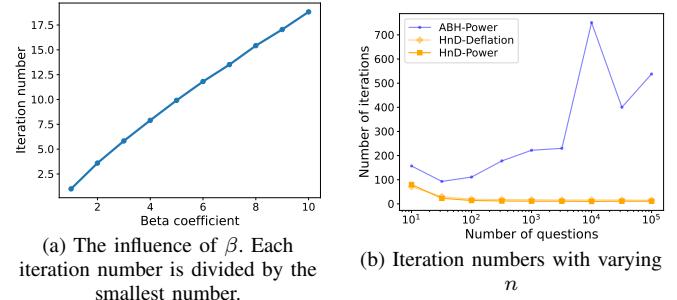


Fig. 14: Detailed Analysis on ABH-power

This model is widely used and studied in crowdsourcing applications for homogenous questions [27], [38], [51], [76], and a recent survey [79] on crowdsourcing recommends it as an implementation with little overhead. However, DS is not suited to model a scenario where *every question can be different*, such as in multiple-choice questions testing student understanding. IRT, in contrast to DS, uses one (or more) latent parameters *for each question*.

B. Detailed Analysis on ABH-power

As discussed in Section III-F, it is possible theoretically to implement ABH in a similar way as HND whose time complexity is $\mathcal{O}(mnt)$. However, instead of $\mathcal{O}(mnt)$, the time complexity of ABH-Power is $\mathcal{O}(mnt + m^2t)$, which can be larger than $\mathcal{O}(mnt)$ when $m \gg n$. We start from the algorithm to analyze ABH-Power in detail.

The algorithm of ABH-Power. Algorithm 2 presents our novel approach for implementing ABH which we call ABH-Power. The main difference from Algorithm 1 are Lines 5 and 6. Note that \mathbf{C} is a $(m \times n)$ matrix but both \mathbf{D} and \mathbf{I} are $(m \times m)$ matrices which are much larger than \mathbf{C} when m is much larger than n . In this case, ABH-Power has time complexity $\mathcal{O}(m^2t)$, which corresponds to the quadratic execution time in Figure 5a. However, this does not explain why ABH-Power is also slower in Figure 5b. Besides, we observe that when question number is large (e.g., 10000) and user number is small (e.g., 100), each iteration of ABH-Power and HND takes similar time. We believe that the reason for ABH-Power being slower also in Figure 5b is due to the required number of iterations until convergence. We discuss this next:

The influence of β . We discussed in Section III-E the comparison between HND and ABH. To implement ABH-power, we need to rely on the matrix $\beta \mathbf{I}_{m-1} - \mathbf{M}$, and β needs to be no smaller than all the entries and eigenvalues of \mathbf{M} . In practice, we set β to be the largest entry in the diagonal matrix of \mathbf{CC}^\top to fulfill the requirements. Therefore, in practice, β is very large. Figure 14a shows an interesting study on the choice of β , which shows that the iteration number of ABH-power is linear in terms of β , which means that the large beta we use results in more iterations compared to HND.

Number of iterations. The above analysis shows us why ABH can be much slower than HND. However, to fully

Algorithm 2: ABH-Power

Input: Response matrix C , randomly initialized user scores s_0

Output: User scores s

```
1:  $s^{\text{diff}} \leftarrow s_0^{\text{diff}}$  // initialize user score differences
2: repeat
3:    $s \leftarrow Ts^{\text{diff}}$  // update user scores
4:    $w \leftarrow C^\top s$  // update option weights
5:    $s \leftarrow Ds - Cw$  // update user scores
6:    $s^{\text{diff}} \leftarrow \beta Is^{\text{diff}} - Ss$  // update user score differences
7:   Normalize  $s^{\text{diff}}$  to be a unit vector
8: until convergence or iteration limit
9:  $s \leftarrow Ts^{\text{diff}}$ 
```

understand why in both [Figure 5b](#), it seems ABH-Power takes longer than linear execution time, we need more detailed evidence of the number of iterations. [Figure 14b](#) shows the numbers of iterations against varying number of users and questions respectively, corresponding to [Figure 5b](#). Each shown data point is also the median iteration number over the 5 runs. [Figure 14b](#) shows that generally as the number of questions increase, the number of iterations increase as well. This explains why in [Figure 5b](#), ABH-Power is not linear.

APPENDIX F REPRODUCIBILITY

In this subsection, we briefly discuss our code repository structure on github [9] (see [Figure 15](#)). A more detailed introduction can be found in the *Readme.md* file in the repository.

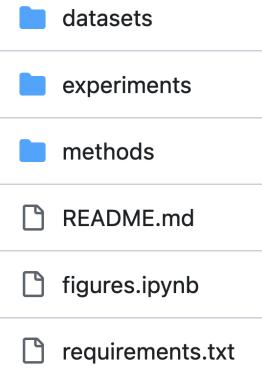


Fig. 15: Repository structure

The *methods* folder contains all the methods including various implementations of HITSnDIFFs and ABH, majority vote, “True-Answer” baseline, HITS-based approaches and GRM-estimator using the girth package. The *experiments* folder contains all the experiments reported in the paper including various accuracy and efficiency experiments on synthetic datasets, experiments on real-world datasets and experiments to compare HnD and ABH. The *datasets* folder contains the real-world datasets we use in the experiments. *Readme.md* describes the repository structure. With the help of *requirements.txt*, all necessary python packages can be

installed. *figures.ipynb* provides a notebook to reproduce every figure of experimental results in this paper.