PARWiS: Winner determination from Active Pairwise Comparisons under a Shoestring Budget

DEV YASHPAL SHETH, Robert Bosch Center for Data Science and AI, Indian Institute of Technology Madras, India

ARUN RAJKUMAR, Robert Bosch Center for Data Science and AI, Indian Institute of Technology Madras, India

We consider the problem of determining a winner among a set of n items by actively comparing pairs of items. We focus on a practical scenario where we are given a *shoestring* budget (say cn for a small c such as 2 or 3) where the usual sample complexity bounds for winner recovery become useless. We focus on the Bradley-Terry-Luce model for noisy comparisons and propose PARWiS, a novel algorithm for Pairwise Active Recovery of Winner under a Shoestring budget. Our algorithm is based on an active version of a spectral ranking procedure combined with a pair selection procedure. We consider several natural disruption measures for the pair selection procedure and show that they are theoretically equivalent to moving along certain gradient directions of popular estimation objectives. We perform extensive synthetic and real-world experiments to show that our algorithm recovers the item at the top effectively with shoestring budgets and outperforms several state-of-the-art algorithms.

Additional Key Words and Phrases: Active learning; Ranking; Pairwise comparisons; Shoestring budget; Winner recovery

ACM Reference Format:
Dev Yashpal Sheth and Arun Rajkumar. 2021. PARWiS: Winner determination from Active Pairwise Comparisons under a Shoestring Budget. In *Workshop on Online and Adaptive Recommender Systems (OARS), 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Singapore.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Ranking from pairwise comparisons is an important problem that has applications in crowd-sourcing [12], social choice and social science [9, 32], recommender systems [18], sports competitions [14] and so on [5, 6, 22, 29, 31]. In the *active* version of the problem, one chooses to query a pair for comparison based on outcomes of comparisons seen so far. The goal typically is to use as few comparisons as possible to recover either a complete ranking of n items or the top-K items or, as we focus in this work, simply recovering the *winner* among the n items. Recovering the winner in the noise free case is an easy problem as one can start with an arbitrary pair and keep promoting the winner, which requires just $n-1$ comparisons. In practice, however, comparisons are seldom noise-free and one usually assumes a reasonable noise model for comparisons. One such popular model is the Bradley–Terry–Luce (BTL) model [8, 25] which assigns probability for items winning in pairwise contests against each other using a latent real valued positive score for each item. While several algorithms have been proposed to recover the winner at the top from noisy pairwise-comparisons under the BTL model, they typically require a sample complexity that is $O(n)$ or $O(n \log \log n)$ with large constants hiding inside the order notation. These constants are problem specific in that they depend on the BTL scores of these items and are usually inevitable for a high confidence winner recovery. Our interest in this paper stems from a practical consideration where such large constants are prohibitive. In particular, we are given a *shoestring* comparison budget B

Manuscript submitted to ACM

to work with and wish to develop an active ranking procedure that recovers the top item under the given budget. A budget less than $n-1$ is hopeless for winner recovery even in the noise-free case, so we will work with budgets to be cn for typically small c such as 2 or 3. Such a budget might be imposed due to several real-world constraints where one might be interested in getting a quick sense of who the winner is among a set of items using very few comparisons.

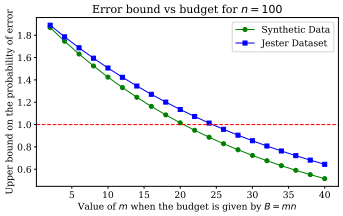

Fig. 1. Plot illustrating that the theoretical upper bound of probability of error for the SELECT algorithm [27] is vacuous for shoestring budgets. The error bound from [27] is $\log n^{-\epsilon}$ where $\epsilon = \frac{2m\Delta}{\ln 2 \log \log n} - 1$ when the budget B is mn. Here we plot the upper bound for our Synthetic case and the Jester dataset [15]. Further details regarding the data can be found in the Experiments Section.

Existing algorithms provide a theoretical sample complexity guarantee for recovering the winner at the top with a confidence parameter at least $1-\delta$. It might seem at first glance that these can easily be converted into a theoretical guarantee on the confidence parameter for any fixed budget B. However, for several state-of-the-art algorithms, there are a couple of pitfalls to watch out for. In some cases, the algorithms are designed such that they need to make at least a certain basic number of comparisons before the theoretical guarantees become valid (For example, Quicksort based Multisort [26]). In the shoestring case, we may not even have these comparisons and hence it is not even clear how to run these algorithms. In some other cases, the theoretical bounds for the shoestring budget may result in vacuous statements such as 'The probability of recovery at the top is less than or equal to 1.5' (Refer Figure 1 for a toy as well as a real-world example). While these bounds are useful to understand the scaling properties and the algorithms may in fact perform better in practice than what the upper bounds guarantee, it is not known how exactly these algorithms will actually perform in the shoestring case. As we will see later, the algorithms we develop in this paper outperform state-of-the-art algorithms in the shoestring case. In this work, we will not be interested in high confidence recovery as it is usually not information-theoretically feasible in the budgets we work with. Instead, we will be interested *practical recovery fractions* i.e., the fraction of times an algorithm recovers the winner with a given shoestring budget in practice.

Based on the above considerations, our goal is in this paper is to develop an active learning algorithm for the BTL noise model that is well suited for shoestring budgets. We also want our algorithm to not have any restrictions on the number of comparisons needed to run as long as the absolute minimum requirement of n comparisons is met.

Our Contributions:

(1) We propose PARWiS, a novel algorithm for Pairwise Active Recovery of Winner under a Shoestring budget. Our algorithm builds upon a spectral ranking procedure and picks the next pair of items to compare based on what we call the 'most disruptive pair' condition.

(2) We theoretically show that several natural conditions to choose the next pair correspond to moving along a weighted averaged gradient step for certain estimation objectives.

(3) We conduct extensive experiments to compare PARWiS with several algorithms. Our results show that PARWiS significantly outperforms the state-of-the-art in recovering the winner on both synthetic and real-world datasets.

2 RELATED WORK

Passive Learning: Rank Centrality [28] is a spectral ranking procedure to recover the true underlying BTL scores of items from a given set of pairwise comparisons. We discuss this algorithm in detail in the Problem Setting and Preliminaries section. The Spectral MLE [13] procedure successively refines the estimates obtained from Rank Centrality [28] via coordinate-wise MLE to determine the top-K items in a given set. A recent paper proposes an accelerated version of the spectral ranking procedure [4] which is orders of magnitude faster than the original. We leave the use of this algorithm for our setting for future works and focus on [28] for the remainder of this paper.

Active Learning: The inception of active ranking from pairwise comparisons came with [19] where the authors consider each item as an embedding in a d-dimensional Euclidean space and use fundamental geometric ideas to speed-up the ranking procedure. The paper proves query complexity bounds for the perfect annotation case as well as the simple noise model of [10] but does not work in the BTL model which is of interest to us. Another work to note is [1] which deals with the general problem of identifying the k most biased coins and shows that $\Theta(\log^* n)$ rounds of adaptivity are enough to achieve worst-case optimal query complexity. The primary focus of the paper is to reduce the rounds of adaptivity however in our setting you can have as many rounds as the budget.

The state-of-the-art in this field is represented by [17, 26, 27]. The Multisort [26] aggregates ranks over multiple Quicksorts via Copeland Aggregation [30] under the BTL assumption. The Active Ranking (AR) [17] algorithm proposes a generic framework where full ranking and top-K or winner determination are special cases. The algorithm maintains confidence scores for items based on which the full set is partitioned into different bins as per requirements. The SELECT [27] is a single elimination tournament where at each layer items are paired in a random fashion and the winner is promoted to the next layer. The algorithm is extended to determine the top-K of n items by running SELECT on subsets of n/K each and then forming a max-HEAP using the K winners. We compare the performance of our algorithm to the above three in the Experiments section.

Bandits: The Relative UCB [34] algorithm was proposed for the dueling bandits problem which is the same as winner determination using active pairwise comparisons. The algorithm extends the UCB idea by maintaining confidence bounds over the standard pairwise comparison matrix. We also consider the successive elimination based Sparse Borda Algorithm [20] which in addition to maintaining confidence bounds over the scores of items, checks for a sparsity condition in the pairwise comparison matrix to eliminate items. Both of the algorithms concern with determining the Borda winner which is also the BTL winner and hence we compare their performance against our algorithm in the shoestring case. A survey of various approaches in the dueling bandits framework can be found in [33].

Crowdsourcing: The Crowd-BT [12] algorithm extends the BTL model by incorporating malicious user behaviour. The algorithm assumes a gaussian prior over the scores of items and a beta prior over the nature of voters and iteratively picks the best possible triplet - a voter to vote for a pair of items - to maximize the global likelihood function. The focus of [12] is to model the voter behaviour and output a robust ranking while we are primarily concerned with winner determination. A recent paper considers the problem of identifying the top-K items in an online crowd-sourced fashion [24]. However, their algorithm is not designed for the BTL model and works with the noisy permutation model of [10]. Finally, two papers [2, 3] introduce the idea of adversarial corruptions. The authors [2] propose a corruption model for spectral ranking algorithms and provide theoretical conditions and an algorithm for recovery while [3] introduce malicious voters to bandits and give a robust elimination style algorithm with optimal regret bounds.

3 PROBLEM SETTING AND PRELIMINARIES

Consider a set of n items for which one is allowed to make pairwise comparison queries in an active fashion. Whenever a pair of items (i, j) is chosen as a query to be compared, either item i or item j is declared the winner. This declaration is made independently of everything else by observing the outcome of a Bernoulli trial with probability P_{ij} of choosing i as the winner. We assume that these noise probabilities arise out of the Bradley-Terry-Luce Model which posits that there exists a score vector $\mathbf{w} \in \mathbb{R}_+^n$ such that $P_{ij} = \frac{w_i}{w_i + w_j}$ for all i, j. A budget B constraining the number of queries one is allowed to perform is provided apriori. The goal of an algorithm is to make at most B comparisons and output the winner among the set of n items. In the BTL case, the winner is simply the item i with the highest BTL score w_i. If there are multiple winners, then recovering any one of them is considered a success for the algorithm. The algorithm will be primarily judged based on it's *recovery fraction* i.e., the fraction of times it recovers the winner over a set of experimental runs. This is a practical proxy for the error probability of the algorithm. In the experiments section, we will elaborate further on other performance metrics which are suited when the scores of the winner and the item at the second position are close to each other.

3.1 Spectral Ranking Algorithm

A popular algorithm for ranking from pairwise comparisons under the BTL model is the spectral ranking algorithm (Algorithm 1). The algorithm when provided with a bunch of pairwise comparisons along with their outcomes, constructs an aperiodic irreducible Markov chain out of it. An approximation for the original BTL score vector is output by computing the stationary distribution of the constructed Markov chain. This algorithm is guaranteed to recover an approximate BTL score vector with a reasonable error, that goes down as n^{-k} for some problem specific constant k, using only $O(n \log n)$ *passive* comparisons chosen uniformly at random. In general for a given set of pairwise comparisons where the pairs are not necessarily chosen at random, the quality of the recovered score vector depends on the second smallest eigenvalue of Laplacian of the associated item comparison graph[1]. To the best of our knowledge, the above spectral ranking algorithm has not been used in the active ranking setting.

Algorithm 1: Spectral Ranking [28]

1: **Require: Graph** $G = ([n], E)$**, pairwise comparison count matrix A**
2: Compute the augmented markov chain $\mathbf{Q}$ from $\mathbf{A}$ as follows

$$Q_{ij} = \begin{cases} \frac{1}{d_{\max}} A_{ij} & \text{if } i \neq j \\ 1 - \sum_{i \neq j} Q_{ij} & \text{if } i = j \end{cases}$$

3: Compute the stationary distribution π of $\mathbf{Q}$ as the limit of

$$p_{t+1}^T = p_t^T \mathbf{Q}$$

4: **Output ranking** σ **by sorting the values of** π

3.2 Incremental Markov Chain Updates

In several practical Markov chain based algorithms, one needs to keep track of the stationary distribution for a time varying Markov chain. Here at every time instant a few edges get added or deleted. Instead of recomputing the stationary

[1] An undirected graph $G = ([n], E)$ where the nodes correspond to the n items and an edge $(i, j) \in E$ iff the pair (i, j) was compared at least once.

distribution from scratch, one typically resorts to performing an incremental update. This is a particularly attractive solution when the number of edge additions or deletions is small. We now present a result from [23] which gives an analytical update rule for computing the stationary distribution of an updated Markov chain using the stationary distribution of a given Markov chain in terms of the edge additions/deletions.

Theorem 1 ([23]). *Let $\boldsymbol{Q}$ be the transition probability of an irreducible Markov chain and suppose that the ith row $\boldsymbol{q}^T$ of $\boldsymbol{Q}$ is updated to produce $\boldsymbol{p}^T = \boldsymbol{q}^T - \delta^T$, the i-th row of $\boldsymbol{P}$, which is also the transition probability of an irreducible chain. If π^T and ϕ^T denote the stationary distributions of $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively, and if $\boldsymbol{A} = \mathbf{I} - \boldsymbol{Q}$, then $\pi^T = \phi^T - \epsilon^T$, where*

$$\epsilon^T = \left[\frac{\phi_i}{1 + \delta^T A^{\#}_{*i}}\right] \delta^T \boldsymbol{A}^{\#}$$

where $A^{\#}_{*i}$ is the $i-$th column of $\boldsymbol{A}^{\#}$ and $\boldsymbol{A}^{\#}$ is the group inverse of $\boldsymbol{A}$.

The above theorem gives a way to incrementally update the stationary distribution of a Markov chain when only one row of the transition probability matrix is affected. In cases where multiple rows are affected, the Theorem can be invoked multiple times in a sequential fashion to get an exact analytical solution of the updated Markov chain. While there are ways to make this update even faster using approximate methods [23], we leave it to future work and focus only on the exact update rule in this work.

4 PARWIS: WINNER RECOVERY ALGORITHM

In this section, we present our proposed algorithm PARWiS for recovering the winner under the BTL model. A pseudo code of our algorithm is shown in Algorithm 2. The algorithm is divided into two phases and we describe these phases separately.

4.1 Phase 1 - Initialization

In the first phase of PARWiS, we use up $n-1$ comparisons out of the B comparisons that we are allowed to use. These $n-1$ comparisons serve as the initial set of comparisons which are used to construct a Markov chain to be used by Phase 2 of the Algorithm. The $n-1$ comparisons are chosen as follows: A permutation $\sigma \in \mathcal{S}_n$ is chosen uniformly at random from the set of all permutations. In the first step $(\sigma(1), \sigma(2))$ is compared. The winner is then compared to $\sigma(3)$ and the algorithm proceeds by promoting the winner of the k-th comparison to be compared with $\sigma(k+1)$ until $n-1$ comparisons are performed.

Phase 1 proceeds by computing the *augmented* Markov chain associated with the current set of $n-1$ comparisons along with $2n$ additional *phantom* comparisons. These comparisons are not used from the budget but are artificially created by introducing a phantom item 0 which is assumed to have 2 phantom comparisons with each of the n items with 1 win and 1 loss against each. It can be shown that this phantom node does not affect the relative ordering of stationary distribution values for the rest of the n items. The advantage of introducing this item is that it serves as a regularizer (or a Bayesian prior). Specifically, by ensuring that the directed comparison graph on the $n+1$ items is strongly connected, it ensures that no item gets a 0 score in the stationary distribution.

4.2 Phase 2- Markov Chain Updates

Phase 2 of PARWiS begins by using the stationary distribution of the computed Markov chain in Phase 1 to select a new pair to be compared. This is done by choosing what we call the *most disruptive pair*. Intuitively, we would like to

Algorithm 2: PARWiS - Winner Recovery Algorithm

Input: Number of items n, Budget $B \geq n-1$ a ranking objective function f;

Phase 1: Initialization $(n-1$ **comparisons)**

- Compare an arbitrary pair, keep promoting winner to the next comparison against an item not compared so far with the winner.
- Using the above $n-1$ comparisons, compute the *Augmented* Markov chain $\mathbf{Q}$ and stationary distribution π as per Algorithm 1

Phase 2: Markov Chain Updates ($(B-n+1)$ **comparisons)** ;

while *Comparisons* $\leq B$ **do**

Compute *most disruptive pair* (i, j) as following where S is the dataset of pairs queried so far,
$i^* = \arg\max_i \pi_i$;

$$j^* = \arg\max_j d_f^S(i^*, j)$$

Query (i^*, j^*), obtain outcome $y \in \{0, 1\}$ and update Markov Chain $\mathbf{Q}$ accordingly;
Incrementally update stationary distribution π using Theorem 1 Update dataset S to $S \cup (i^*, j^*, y)$;

end

Declare $i^* = \arg\max_i \pi_i$ as the winner

compare the current winner (item with the highest stationary distribution value) with an item which has the highest chance of disrupting the winner. A natural approach here would be choose the item with the second highest score to compare against the winner. However, this may not be the ideal approach in the shoestring budget case for two reasons:

(i) **Local Effect:** Several items may not have been compared enough number of times against the current winner and this captures the local effect of a pair of items

(ii) **Global Effect:** Even if a pair has been compared several times with the winner, because of other pairs not being compared enough number of times, there might be discrepancy between the empirical probability of a pair and the probability computed using the stationary distribution values i.e., $\hat{P}_{ij}$ may not be representative of $\frac{\pi_i}{\pi_i+\pi_j}$. This captures the global effect of the comparisons on a given pair.

Thus we would like to come up with a principled way of choosing the most disruptive pair taking into account the above considerations.

4.3 Weighted Average Gradient Norm Criterion

We now describe our proposed method to pick the most disruptive pair given that a Markov chain has been constructed in Phase 1 of the algorithm. Let π be the stationary distribution at the end of Phase 1. Assume without loss of generality that $\pi_1 \geq \pi_2 \geq \ldots \pi_n$. If the pair $(1, j)$ is chosen for comparison, either 1 will be preferred over j or j over 1. The current probability for the former event is $\frac{\pi_1}{\pi_1+\pi_j}$ and the latter is $\frac{\pi_j}{\pi_1+\pi_j}$. For each of the events, we could re-run a ranking algorithm with the extra comparison. However, our goal is not to obtain a ranking but to understand the direction in which the ranking moves for each of these outcomes. We formalize this by taking a step from the current estimate in the direction of the gradient of a suitable ranking objective (log likelihood or Markov chain objective or least squares) and computing the length of the gradient. The pair $(1, j)$ for which the weighted length of the gradient, weighted by the probability of the two events is highest, will be chosen as the next pair for comparison. We formalize this below:

Let $f^{LogL}(\theta; S)$ denote the log-likelihood function for the parameter θ of interest (BTL scores or an appropriate monotonic function of them, for instance log scores) given a dataset S. Let $f^{LS}(\theta; S)$ denote the least squares objective function for the parameter θ and let $f^{Markov}(\theta; S)$ denote the objective function that computes the stationary distribution of a Markov chain constructed from S. A triplet $(i, j, y) \in S$ would mean that the pair (i, j) was compared and $y = 1$ indicates i was preferred to j and $y = 0$ indicates otherwise. Let $S^1_{ij} = S \cup \{(i, j, 1)\}$ and $S^0_{ij} = S \cup \{(i, j, 0)\}$. For a given ranking objective f, and a pair (i, j), define the disruption measure at π as follows:

$$d^S_f(i, j) = \frac{\pi_i}{\pi_i + \pi_j} \|\nabla f(\pi; S^1_{ij}))\| + \frac{\pi_j}{\pi_i + \pi_j} \|\nabla f(\pi; S^0_{ij}))\|$$

Theorem 2. *Let $\boldsymbol{Q}$ be the Markov chain constructed using a dataset S and let π be it's stationary distribution. Let N_{ij} be the number of times a pair (i, j) has been compared and let $\hat{P}_{ij}$ be the fraction of times item i is preferred over j in S. Let $\alpha_{ij} = \frac{P_{ij}}{\hat{P}_{ij}}$ measure the discrepancy between the empirical estimate for the pair (i, j) and the estimate obtained using π where $P_{ij} = \frac{\pi_i}{\pi_i + \pi_j}$. Then,*

$$d^S_{f^{LogL}}(i, j) = 2\sqrt{2} \frac{\pi_i \pi_j}{\pi_i + \pi_j} \tag{1}$$

$$d^S_{f^{LS}}(i, j) = \frac{2\sqrt{2}}{\pi_i + \pi_j} \left[\pi_i \log\left(1 + \frac{N_{ij}}{\hat{P}_{ij}}\right) + \pi_j \log\left(1 + \frac{N_{ij}}{\hat{P}_{ji}}\right) \right]$$

$$\simeq \frac{2\sqrt{2}}{N_{ij}} (\alpha_{ij} + \alpha_{ji}) \tag{2}$$

$$d^S_{f^{Markov}}(i, j) = \frac{\sqrt{2}}{n(N_{ij} + 1)} \frac{\pi_i \pi_j}{\pi_i + \pi_j} \left(\frac{1}{\alpha_{ij}} + \frac{1}{\alpha_{ji}} \right) \tag{3}$$

The proof of Theorem 2 has been included in the Appendix. It reveals some interesting insights about our pair selection procedure. In particular, it shows that the log-likelihood objective would always prefer to pick the pair with the highest two scores in the current stationary distribution. As we argued earlier, this may not be the right objective in the shoestring budget case. Interestingly, both the Markov chain objective and the least squares objective attempt to capture the *local* and *global* effects - via selecting pairs which balance between the number of times they have been compared against the top item and how well they approximate the current probability estimates. This is precisely the goal we set out to achieve for pair selection. As we will see, the experimental results confirm that the Markov and LS objectives perform significantly better than the log-likelihood objective.

Reduced search space: In the way we have defined the most disruptive pair above, we can potentially consider computing the disruption score for all $\binom{n}{2}$ pairs in every iteration to decide on the pair to be queried. However for the shoestring case as mentioned earlier, it is beneficial to look at the pairs which include the current winner as is asserted in the Experiments even for a random pair selection strategy. Hence, we reduce this to just considering $n - 1$ pairs which involve the current top item as the goal is to recover the winner as fast as possible.

Runtime: Phase 1 of PARWiS involves running the spectral raking algorithm which can be done in $O(n^2)$. In Phase 2 after every vote collected, the algorithm needs to incrementally update the markov chain and the stationary distribution (scores) which takes $O(n^2)$. Computing the most disruptive pair takes just $O(n)$ computations. The complexity is similar to bandit-style algorithms [20, 34] which involve steps like determining the current Borda winner, updating the confidence scores or checking the sparsity conditions. Algorithms like SELECT [27] and AR [17] do not require such extra computations at every step. However as we will see next, the Phase 2 of our algorithm is crucial to getting SoTA performance in the shoestring regime.

5 EXPERIMENTS

In this section, we present the results of running our algorithm with different disruption measures followed by comparison of our algorithm with the current state-of-the-art on carefully chosen performance metrics. We first identify that the Markov chain objective is the best disruption measure and then compare its performance with the rest of the baseline algorithms in literature. Both sets of experiments are carried out on synthetic as well as real-world datasets.[2]

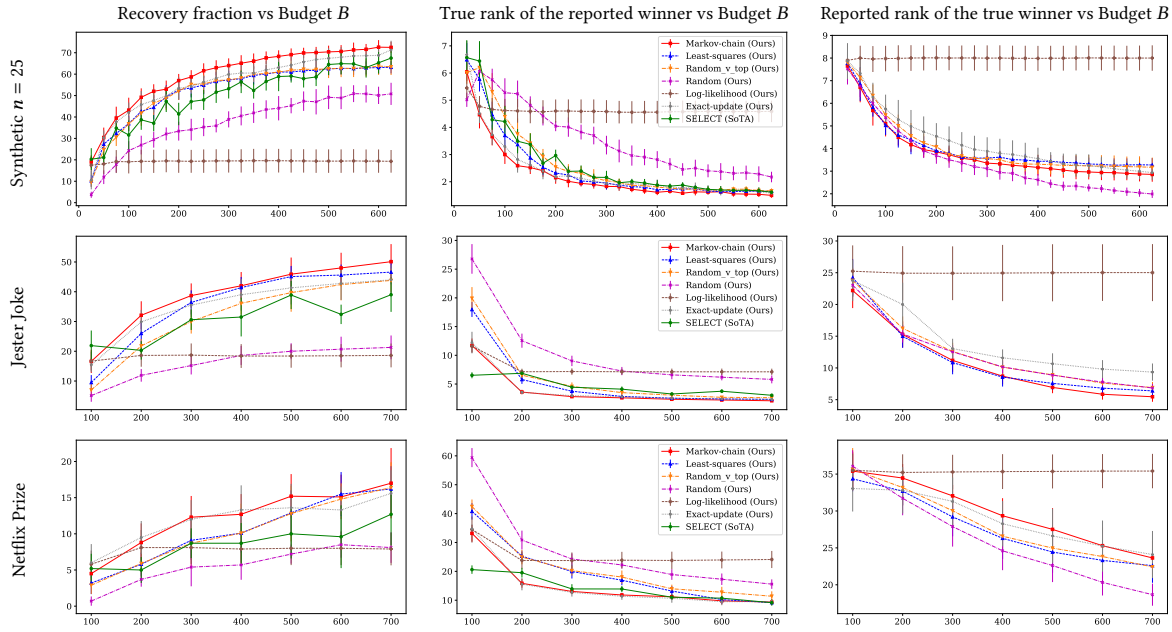

Fig. 2. Performance of PARWiS with different disruption measures and the current SoTA on synthetic and real-world datasets. Each graph shows a performance metric on the y-axis vs the budget B on the x-axis. The dataset used for the particular set of experiments is indicated on the left (row-wise), while the metric plotted is indicated on the top (column-wise). The legends are the same across all figures. The Markov chain objective is the overall best performing disruption measure.

5.1 Performance Metrics

We consider the following performance metrics to judge the quality of the various algorithms.

(a) **Recovery fraction:** For a set of experimental runs, the fraction of times the algorithm succeeds in recovering the true winner (reported as a number out of 100).

(b) **True rank of reported winner:** The average true rank in the BTL ordering of the reported winner over runs. Smaller the value, the better is the performance of the algorithm.

(c) **Reported rank of true winner:** For algorithms which produce a winner by first producing a ranking, the rank of the true winner averaged over runs. This measure can be applied to all algorithms except SELECT.

[2]https://github.com/anonymous-PARWiS/active-raking-under-shoe-string-budget

5.2 Disruption Measures

Apart from the disruption measures for the log-likelihood, least-squares and the Markov chain objectives as defined in Theorem 2, we consider three more cases in order to demonstrate the advantage of the measures. The first case we consider is of *exact-update* d^S_{exact} which is defined as the weighted L2-norm difference between current stationary distribution and the distributions obtained by adding the votes $(i, j, 1)$ and $(i, j, 0)$ to the set of comparisons S given by,

$$d^S_{exact}(i,j) = \frac{\pi_i}{\pi_i + \pi_j}\|\pi - \pi^1_{ij}\|_2 + \frac{\pi_j}{\pi_i + \pi_j}\|\pi - \pi^0_{ij}\|_2$$

The motive behind including d^S_{exact} is to demonstrate that taking a single step in the direction of optimizing the objective function (Markov chain) is more stable than recomputing the entire stationary distribution. The Markov chain in the shoestring cases is inherently random because of the lack of examples seen so far and this randomness is multiplied several times in the new estimate of the stationary distribution. Also, it is computationally more expensive than the simple gradient-based disruption measures.

We also consider the cases of *random* selection from *all* possible pairs and the case of *random vs the current winner* (random_v_top) which pairs the current top with any of the rest $n - 1$ items uniformly at random. As marked earlier, these two cases will serve as a baseline for the disruption measures as well as assert the claim that we need not consider all $\binom{n}{2}$ pairs and it is beneficial to just look at the current winner vs the rest in the shoestring case.

5.3 Baseline Algorithms

We point out that certain algorithms may not be directly applied to this setting and some modifications may be required. The details of the implementation of baseline algorithms and their specific modifications are as follows.

(i) **SELECT:** The simplest algorithm yet the most competitive is the top-selection algorithm of [27]. It runs an elimination tournament on n vertices where the items are initially paired arbitrarily and compared against each other. The winners are promoted to the next level and compared against each other until a single winner is found. When the budget is greater than n (say cn), at each level c comparisons are made for every pair to decide the winner. In cases of ties, the winner is chosen at random.

(ii) **Multisort:** The algorithm [26] runs multiple noisy quicksorts over the set of items and aggregates the rankings using Copeland winner selection rule [30]. In our implementation, at any point if we run out of budget, we complete that particular quicksort using uniformly random comparisons. The choice is made in order to run at least one iteration of the algorithm for budget values like $n, 2n$.

(iii) **Active Ranking (AR):** The AR [17] is a non-parametric algorithm which adapts the bandits-style successive elimination type algorithm to pairwise ranking. The original algorithm does not terminate until a definite winner is selected after refining the confidence bounds. Here we report the item with the highest score at the moment the budget expires.

(iv) **Relative Upper Confidence Bound (RUCB):** This algorithm [34] is designed for the dueling bandits framework with the goal to play the winner as frequently as possible in order to minimize a natural notion of regret. It runs within a given budget and hence no changes are required.

(v) **Successive Elimination with Comparison Sparsity (SECS):** Also known as the Sparse Borda Algorithm [20], it augments a successive-elimination based bandit algorithm with a carefully designed *sparsity* condition. Like AR, the algorithm is stopped as soon as the budget expires and item with the highest score is reported.

It is important to note that AR, RUCB and SECS aren't specifically suited to the Shoestring case as with just $2n$ comparisons, there can be multiple items with the highest score (say ℓ items) including the true winner. In such cases, we report the rank of the true winner as $(\ell + 1)/2$ which is simply the expected rank of an uniformly at random chosen ordering among the ℓ items. We also wish to point out that the algorithms like LambdaMART [11] are not applicable to our setting. There are no feature vectors associated with items and information is only available in the form of actively queried pairwise comparison outputs.

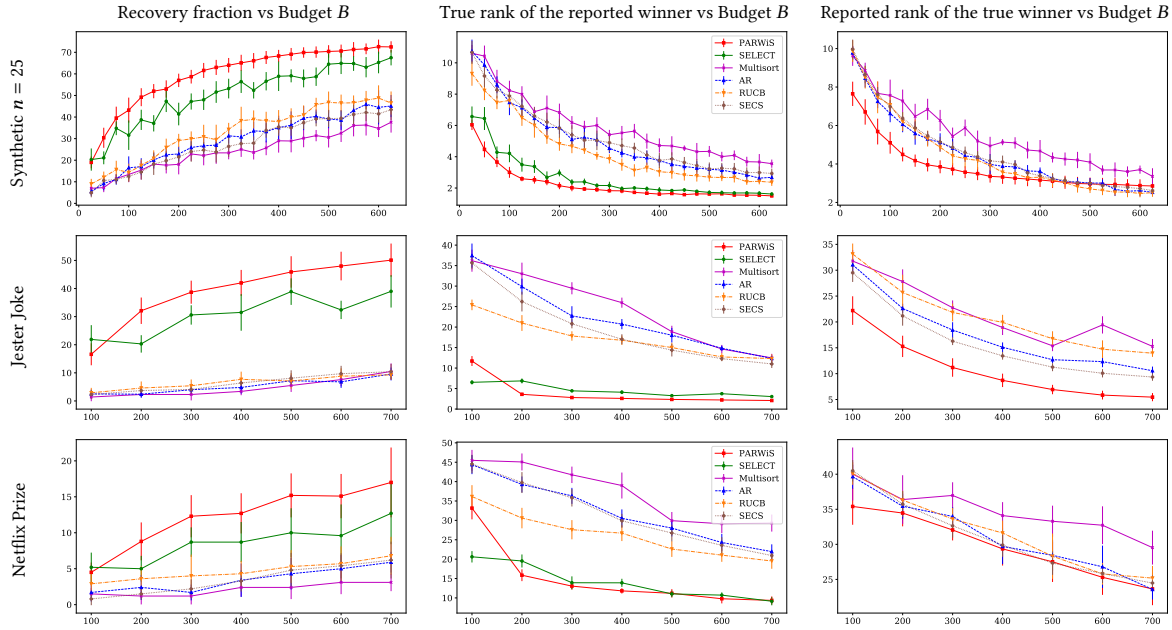

Fig. 3. Performance of PARWiS and several baseline algorithms on synthetic and real-world datasets. Refer 2 caption for layout details. PARWiS consistently outperforms the SoTA algorithms across all datasets with varying degrees of separation ($\Delta_{1,2}$). Refer the Experiments section and *Separation vs Recovery* for more details.

5.4 Synthetic Experiments

We now describe the synthetic experiments performed to evaluate our algorithm against various baselines described earlier. We consider three choices for n namely 10, 25 and 50. In each case, the BTL scores were fixed to be such that the winner gets a score of 100 while the rest of the items get a score sampled and fixed uniformly at random from $[0, 75]$. We fix the separation of winner from the rest for our synthetic experiments as have a variety of real-world datasets as well with varying separations as we will see later. All the metrics are averaged over 10 independent runs of the experiment. The results computing the three performance metrics are shown in Figures 2 and 3. The experiments with $n = 25$ are included in the main text, $n = 10, 50$ can be found in the appendix.

It is easy to see from Figure 2 that the Markov chain objective function is overall the best performing disruption measure closely followed by the least-squares objective. Both of them perform better than the exact-update rule and all three are better than the current state-of-the-art. Fixing one item as the current winner is indeed beneficial than looking all possible pairs as is evident by the random_v_top and random pair selection strategies. The log-likelihood

objective performs the worst as it fixes upon comparing the two items with the highest scores repeatedly and therefore relies more on the performance at the end of Phase 1.

As can be seen in Figure 3, PARWiS clearly outperforms the state-of-the-art algorithms in all metrics reported. The gain is most substantial for the case of Shoestring budget of $2n$. PARWiS not only outperforms in terms of the Recovery Fraction but also has scores low in terms of the other two metrics. This shows that PARWiS is robust and can also be extended to produce the top-K items if necessary.

We note that the performance gain for PARWiS is significant in the Phase 2 of the algorithm where it explicitly takes advantage of the BTL model assumption. This phenomenon is observed across all the settings we present against the SELECT algorithm for all metrics considered. Also, our experimental results assert the claim of [27] that the SELECT converges to the true top item faster than AR i.e. with lesser number of comparisons.

5.5 Real-world Experiments

We next test the performance of PARWiS on real world datasets. We consider the following standard benchmark datasets for our purposes - Sushi-A, Sushi-B [21], Jester [15], Movielens [16] and Netflix [7]. In the case of Sushi-A/B and Jester, we use the entire dataset while in the case of Movielens and Netflix, we retain 100 movies which have the most number of ratings. In order to obtain a true underlying scores for the items of the Sushi data set, we convert the rankings provided by each user for k Sushi ($k = 10$ for both Sushi-A and Sushi-B) into $\binom{k}{2}$ pairwise comparisons and construct the empirical probability for pair of items. We run the spectral ranking algorithm on this matrix to obtain scores for these items. We then use these scores as the true scores for all the active learning algorithms. In the case of Jester, Movielens and Netflix datasets, we convert the ratings into pairwise probabilities by first computing the average difference in ratings between the pair of movies across users and then using a logistic function to convert it into probabilities. Once the pairwise probabilities are available, we convert them into true scores similar to the Sushi dataset. We present the results of Jester and Netflix in the main-text while the rest are included in the appendix.

Our results for the various algorithms for these datasets under different performance metrics are shown in Figures 2, 3. The disruption measures follow the same expected trend as in the synthetic case. The Markov chain objective is the overall best closely followed by least-squares and the exact-update. As can be seen, our algorithm with the Markov chain objective outperforms the rest of the baselines in all the metrics across all datasets and budgets. We again observe the crucial performance gain due to the Phase 2 of our algorithm as we move from $B = n$ to $2n$.

Separation vs Recovery: The complexity bounds of the recovery at the top for baseline algorithms like SELECT [27] and AR [17] are proportional to the inverse of the separation $\Delta_{1,2}$. Here $\Delta_{1,2}$ is defined as $(P_{1,2} - 0.5)^2$ where $1, 2$ denote the items ranked 1 and 2 according to the true underlying BTL score vector w. The delta value for our synthetic experiments is ~ 0.0051 while for the various datasets are Sushi-A (0.0292), Sushi-B (0.00004), MovieLens (0.00006), Jester (0.0043) and Netflix (0.0002). As can be seen in Figure 3 and Figure 4 in the Appendix, our algorithm beats the current state-of-the-art for all delta values and the gain is more significant when the datasets get tougher (when the delta value decreases).

Datasets and the BTL model: Note that if the pairwise probabilities obtained from the comparisons indeed follow a true BTL model, then the spectral ranking algorithm will recover these true BTL scores and we have a clear winner to compare our algorithms against. However, in real-world datasets the probabilities will not always adhere to a BTL model. We thus find an approximate BTL model using the spectral ranking algorithm. One can also choose to do a maximum likelihood estimate for the BTL scores or do a least squares fit by projecting the comparison probability matrices on the set of all BTL induced comparison matrices [29]. In practice, these methods produce by and large similar

rankings and hence we choose the spectral ranking algorithm as our choice of converting comparison probabilities to scores.

Choice of B: We wish to point out that even though PARWiS was designed with the shoestring budget in mind, it consistently performs well even for larger budgets. In the shoestring case, the improvement starts right from budgets equal to $2n$. In our experiments, we increase the budgets in steps of n so that SELECT algorithm can be implemented in a clean fashion (otherwise, one needs to add too many non-informative, 0.5 probability coin toss, comparisons with for these algorithms). However, for PARWiS there is no restriction of the budget to be of the form cn for some c.

6 CONCLUSION

In this paper, we introduce PARWiS - a simple and intuitive algorithm designed to produce a winner using active pairwise comparisons under the BTL noise model. We propose a framework to actively choose pairs using disruption measures that use weighted gradients and show that our choices lead to good measures for popular ranking objectives. Our algorithm significantly outperforms existing algorithms even under shoestring budgets. We wish to explore top-K recovery in the future in addition to understanding the theoretical properties of PARWiS.

REFERENCES

[1] A. Agarwal, S. Agarwal, S. Assadi, and S. Khanna. 2017. Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons. In *Proceedings of the 2017 Conference on Learning Theory*, Vol. 65. 39–75.

[2] A. Agarwal, S. Agarwal, S. Khanna, and P. Patil. 2020. Rank Aggregation from Pairwise Comparisons in the Presence of Adversarial Corruptions. In *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119. 85–95.

[3] A. Agarwal, S. Agarwal, and P. Patil. 2021. Stochastic Dueling Bandits with Adversarial Corruption. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, Vol. 132. 217–248.

[4] A. Agarwal, P. Patil, and S. Agarwal. 2018. Accelerated Spectral Ranking. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. 70–79.

[5] N. Ailon. 2012. An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity. *Journal of Machine Learning Research* 13, 5 (2012), 137–164.

[6] N. Alion, M. Charikar, and A. Newman. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM* 55, 5 (2008).

[7] J. Bennett and S. Lanning. 2007. The Netflix Prize. In *Proceedings of KDD Cup and Workshop*.

[8] R. A. Bradley and M. E. Terry. 1952. Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons. *Biometrika* 39, 3-4 (1952), 324—-345.

[9] F. Brandt, V. Conitzer, and U. Endriss. 2012. Computational social choice. *Multiagent Systems* (2012), 213–283.

[10] M. Braverman and E. Mossel. 2008. Noisy sorting without resampling. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. 268—-276.

[11] C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview.

[12] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 193–202.

[13] Y. Chen and C. Suh. 2015. Spectral MLE: top-K rank aggregation from pairwise comparisons. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37. 371—-380.

[14] A. E. Elo. 1978. *The Rating Of Chess Players, Past and Present.* Arco Pub., New York.

[15] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. 2001. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval* 4, 2 (2001), 133–151.

[16] F. Maxwell Harper and J. A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015).

[17] R. Heckel, N. B. Shah, K. Ramchandran, and M. J. Wainwright. 2019. Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics* 47, 6 (2019), 3099–3126.

[18] N. Houlsby, F. Huszár, Z. Ghahramani, and J. M. Hernández lobato. 2012. Collaborative Gaussian processes for preference learning. *Advances in Neural Information Processing Systems* 25 (2012), 2096–2104.

[19] K. G. Jamieson and R. D. Nowak. 2011. Active Ranking using Pairwise Comparisons. *Advances in Neural Information Processing Systems* 24 (2011), 2240–2248.

[20] K. Jamiesona, S. Katariya, A. Deshpande, and R. Nowak. 2015. Sparse Dueling Bandits. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, Vol. 38. 416–424.
[21] T. Kamishima. 2003. Nantonac collaborative filtering:Recommendation based on order responses. In *Proceedings of The ninth International Conference on Knowledge Discovery and Data Mining*. 583—-588.
[22] C. Kenyon-Mathieu and W. Schudy. 2007. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 95–103.
[23] A. N. Langville and C. D. Meyer. 2006. Updating Markov Chains with an Eye on Google's PageRank. *SIAM J. Matrix Anal. Appl.* 27, 4 (2006), 968—-987.
[24] S. Liang and L. de Alfaro. 2020. Online Top-K Selection in Crowdsourcing Environments. In *Proceedings of the 6th International Conference on Computing and Data Engineering*. 252–259.
[25] R. D. Luce. 1959. *Individual Choice Behavior: A Theoretical Analysis*. Wiley.
[26] L. Maystre and M Grossglauser. 2017. Just Sort It! A Simple and Effective Approach to Active Preference Learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 2344—-2353.
[27] S. Mohajer, C. Suh, and A. Elmahdy. 2017. Active learning for top-K rank aggregation from Noisy comparisons. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 2488—-2497.
[28] S. Negahban., S. Oh, and D. Shah. 2016. Rank Centrality: Ranking from Pairwise comparisons. *Operations Research* 65, 1 (2016), 266–287.
[29] A. Rajkumar and S. Agarwal. 2014. A Statistical Convergence Perspective of Algorithms for Rank Aggregation from Pairwise Data. In *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32(1). 118–126.
[30] D. G. Saari and V. R. Merlin. 1996. The Copeland method. *Economic Theory* 8 (1996), 51–76.
[31] T. L. Saaty. 2008. Decision making with the analytic hierarchy process. *International Journal of Services Sciences* 1, 1 (2008), 83–98.
[32] M. J. Salganik and K. E. C. Levy. 2015. Wiki Surveys: Open and Quantifiable Social Data Collection. *PLoS ONE* 10, 5 (2015).
[33] Y. Sui, M. Zoghi, K. Hofmann, and Y. Yue. 2018. Advancements in Dueling Bandits. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. 5502–5510.
[34] M. Zoghi, S. Whiteson, R. Munos, and M. de Rijke. 2014. Relative Upper Confidence Bound for the K-Armed Dueling Bandit Problem. In *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32. II–10—-II–18.

A PROOF OF THEOREM 2

Apart from the notation used above, let π^1_{ij} and $\mathbf{Q}^1_{ij}$ as the stationary distribution and Markov chain corresponding to S^1_{ij}. Let θ be a vector such that $\forall_i \pi_i = e^{\theta_i}$. Finally, let N^1_{ij} denote the number of times i has won over j and therefore $\hat{P}_{ij} = \frac{N^1_{ij}}{N_{ij}}$.

(1) Log-likelihood objective:

$$f^{LogL}(\theta, S) = \Pi_{(i,j)\in S} \left(\frac{e^{\theta_i}}{e^{\theta_i} + e^{\theta_j}}\right)^{N^1_{ij}} \left(\frac{e^{\theta_j}}{e^{\theta_i} + e^{\theta_j}}\right)^{N^0_{ij}}$$
$$= \Pi_{(i,j)\in S} [N^1_{ij}\theta_i + N^0_{ij}\theta_j - (N_{ij}) \log(e^{\theta_i} + e^{\theta_j})]$$

Assuming that the current θ is optimal w.r.t. the comparisons seen so far S, we have $\forall_i \nabla f^{LogL}(\theta_i, S)) = 0$. Now,

$$\nabla f^{LogL}(\theta_k, S)) = \sum_{i\neq k} N^1_{ki} - (N_{ki}) \frac{e^{\theta_k}}{e^{\theta_k} + e^{\theta_i}}$$

Therefore, when we calculate the gradients w.r.t. the current θ after adding a new comparison $(i, j, 1)$, we have:

$$\nabla f^{LogL}(\theta_i, S^1_{ij})) = \nabla f^{LogL}(\theta_i, S)) + 1 - \frac{e^{\theta_i}}{e^{\theta_i} + e^{\theta_j}} = P_{ji}$$
$$\nabla f^{LogL}(\theta_j, S^1_{ij})) = \nabla f^{LogL}(\theta_i, S)) - \frac{e^{\theta_j}}{e^{\theta_i} + e^{\theta_j}} = -P_{ji}$$
$$\nabla f^{LogL}(\theta_k, S^1_{ij})) = 0 \ \forall_k \text{ s.t. } k \neq i, j$$

We define the disruption measure as follows:

$$\begin{aligned} d^S_{fLogL}(i,j) &= \frac{\pi_i}{\pi_i+\pi_j}\|\nabla f^{LogL}(\theta, S^1_{ij}))\|_2 \\ &\quad + \frac{\pi_j}{\pi_i+\pi_j}\|\nabla f^{LogL}(\theta, S^0_{ij}))\|_2 \\ &= \left(\frac{\pi_i}{\pi_i+\pi_j}\right)\left(\sqrt{2}\frac{\pi_j}{\pi_i+\pi_j}\right) \\ &\quad + \left(\frac{\pi_j}{\pi_i+\pi_j}\right)\left(\sqrt{2}\frac{\pi_i}{\pi_i+\pi_j}\right) \\ &= 2\sqrt{2}\frac{\pi_i\pi_j}{\pi_i+\pi_j} \end{aligned}$$

(2) Least-squares objective:

$$\begin{aligned} f^{LS}(\theta) &= \sum_{(i,j)\in S}[(\theta_i-\theta_j)-(\log(N^1_{ij})-\log(N^0_{ij}))]^2 \\ \nabla f^{LS}(\theta_k, S) &= 2\sum_{i\neq k}[(\theta_k-\theta_i)-(\log(N^1_{ki})-\log(N^0_{ki}))] \\ \nabla f^{LS}(\theta_i, S^1_{ij}) &= \nabla f^{LS}(\theta_i, S) - 2\log(N^1_{ij}) + 2\log(N^1_{ij}+1) \\ &= 2\log\left(1+\frac{1}{N^1_{ij}}\right) \\ \nabla f^{LS}(\theta_j, S^1_{ij}) &= -2\log\left(1+\frac{1}{N^1_{ij}}\right) \end{aligned}$$

$$\begin{aligned} d^S_{fLS}(i,j) &= \frac{\pi_i}{\pi_i+\pi_j}\|\nabla f^{LS}(\theta, S^1_{ij}))\|_2 + \frac{\pi_j}{\pi_i+\pi_j}\|\nabla f^{LS}(\theta, S^0_{ij}))\|_2 \\ &= \frac{\pi_i}{\pi_i+\pi_j}\left(2\sqrt{2}\log\left(1+\frac{1}{N^1_{ij}}\right)\right) \\ &\quad + \frac{\pi_j}{\pi_i+\pi_j}\left(2\sqrt{2}\log\left(1+\frac{1}{N^0_{ij}}\right)\right) \\ &= \frac{2\sqrt{2}}{\pi_i+\pi_j}\left[\pi_i\log\left(1+\frac{1}{N^1_{ij}}\right)+\pi_j\log\left(1+\frac{1}{N^0_{ij}}\right)\right] \\ &\simeq \frac{2\sqrt{2}}{\pi_i+\pi_j}\left(\frac{\pi_i}{N^1_{ij}}+\frac{\pi_j}{N^0_{ij}}\right) = \frac{2\sqrt{2}}{N_{ij}}(\alpha_{ij}+\alpha_{ji}) \end{aligned}$$

Note: For experiments we use the exact expression as calculated above.

(3) Markov chain objective:

$$d^S_{fMarkov}(i,j) = \frac{\pi_i}{\pi_i+\pi_j}\|\pi^T - \pi^T\mathbf{Q}^1_{ij}\|_2 + \frac{\pi_j}{\pi_i+\pi_j}\|\pi^T - \pi^T\mathbf{Q}^0_{ij}\|_2$$

Here, we have $\mathbf{Q}^1_{ij} = \mathbf{Q} + \mathbf{C}^1_{ij}$ where $\mathbf{C}^1_{ij}$ is defined as follows: (refer [28], the chain also accounts for the phantom item 0)

$$C^1_{ij}(r,s) = \begin{cases} \frac{1}{n}\left(\frac{N^0_{ij}}{N_{ij}+1} - \frac{N^0_{ij}}{N_{ij}}\right) & \text{if } r=i \text{ and } s=j \\ -\frac{1}{n}\left(\frac{N^0_{ij}}{N_{ij}+1} - \frac{N^0_{ij}}{N_{ij}}\right) & \text{if } r=s=i \\ \frac{1}{n}\left(\frac{N^1_{ij}+1}{N_{ij}+1} - \frac{N^1_{ij}}{N_{ij}}\right) & \text{if } r=j \text{ and } s=i \\ -\frac{1}{n}\left(\frac{N^1_{ij}+1}{N_{ij}+1} - \frac{N^1_{ij}}{N_{ij}}\right) & \text{if } r=s=j \\ 0 & \text{otherwise} \end{cases}$$

For simplicity, let $\Delta_{ij} = \frac{1}{n}\left(\frac{N^0_{ij}}{N_{ij}+1} - \frac{N^0_{ij}}{N_{ij}}\right) = -\frac{N^0_{ij}}{nN_{ij}(N_{ij}+1)}$ and $\Delta_{ji} = \frac{1}{n}\left(\frac{N^1_{ij}+1}{N_{ij}+1} - \frac{N^1_{ij}}{N_{ij}}\right) = \frac{N^0_{ij}}{nN_{ij}(N_{ij}+1)}$ where both the expressions are simplified by writing $N_{ij} = N^1_{ij} + N^0_{ij}$. Using these, we have,

$$d^S_{fMarkov}(i,j) = \frac{\pi_i}{\pi_i+\pi_j}\|\pi^T \mathbf{C}^1_{ij}\|_2 + \frac{\pi_j}{\pi_i+\pi_j}\|\pi^T \mathbf{C}^0_{ij}\|_2$$

where,

$$\begin{aligned}
\pi_T \mathbf{C}^1_{ij} &= \begin{bmatrix} 0 & \dots & -\pi_i\Delta_{ij} + \pi_j\Delta_{ji} & \dots & \pi_i\Delta_{ij} - \pi_j\Delta_{ji} & \dots \end{bmatrix} \\
\|\pi^T \mathbf{C}^1_{ij}\|_2 &= \sqrt{2}(-\pi_i\Delta_{ij} + \pi_j\Delta_{ji}) \\
&= \sqrt{2}\frac{N^0_{ij}(\pi_i+\pi_j)}{nN_{ij}(N_{ij}+1)} = \frac{\sqrt{2}\pi_j}{n(N_{ij}+1)\alpha_{ji}}
\end{aligned}$$

Therefore, the disruption measure is given by:

$$d^S_{fMarkov}(i,j) = \frac{\sqrt{2}}{n(N_{ij}+1)}\frac{\pi_i\pi_j}{\pi_i+\pi_j}\left(\frac{1}{\alpha_{ij}} + \frac{1}{\alpha_{ji}}\right)$$

Note: In order to avoid arithmetic errors in the Least-squares and the markov chain objectives we add a regularizing constant $\epsilon = 1\ \forall_{i\neq j} N^b_{ij}$ where $b \in \{0,1\}$.

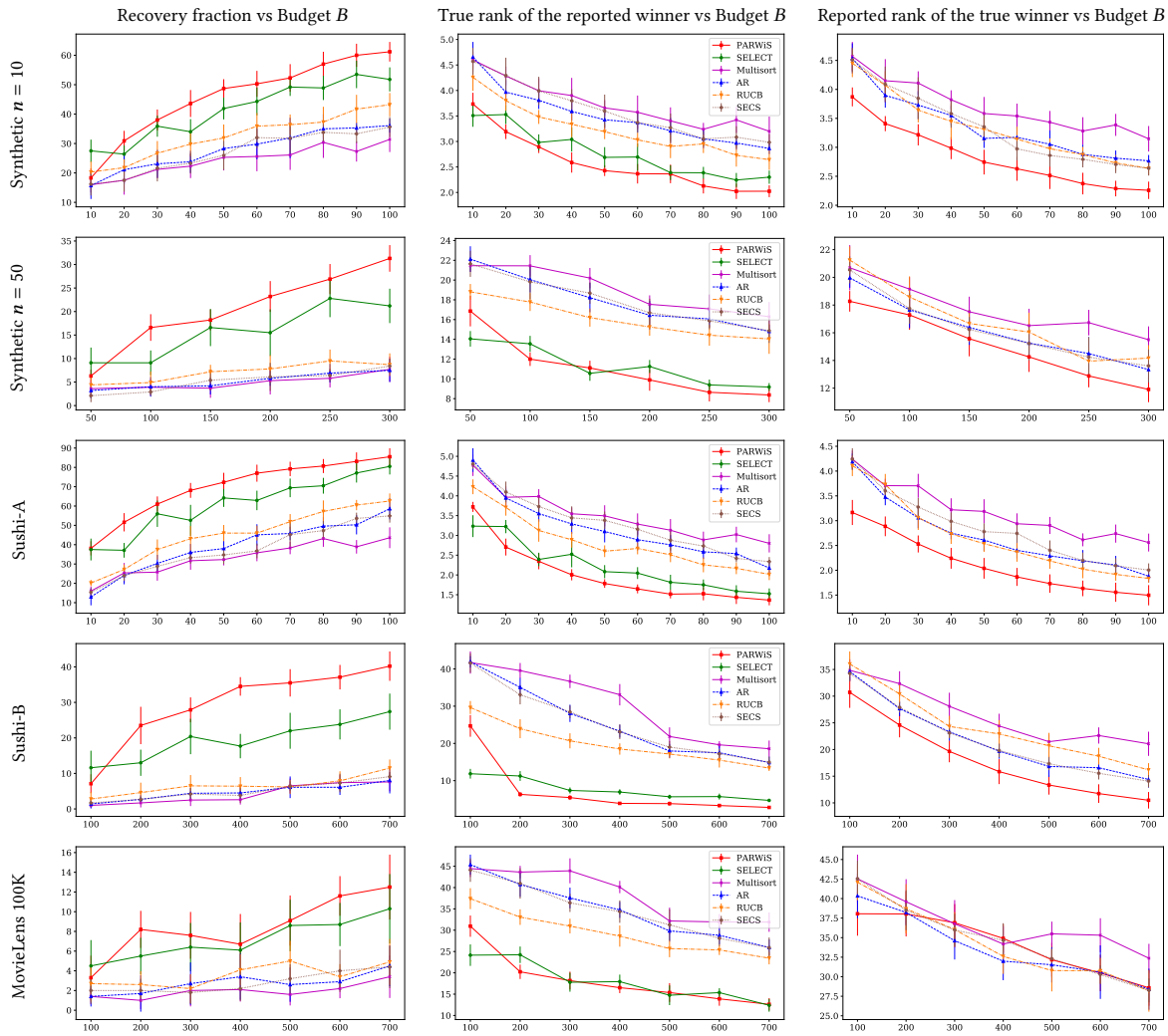

Fig. 4. Performance of PARWiS and several baseline algorithms on synthetic and real-world datasets. Each graph shows a performance metric on the y-axis vs the budget B on the x-axis. The dataset used for the particular set of experiments is indicated on the left (row-wise), while the metric plotted is indicated on the top (column-wise). The legends are the same across all figures. PARWiS consistently outperforms the SoTA algorithms across all datasets with varying degrees of separation ($\Delta_{1,2}$). Refer Experiments section for details.