FAST MAXIMUM-LIKELIHOOD RANKINGS FROM PAIRWISE PREFERENCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Computing rankings from pairwise (probabilistic) comparisons is ubiquitous in machine-learning pipelines, for example when aggregating LLM-as-a-judge preferences or combining the rankings of multiple voters. Finding the exact maximum-likelihood ranking is NP-hard, so pipelines typically settle for heuristics, compromising quality. We introduce XRank, a fast exact solver that substantially outperforms existing exact methods and makes optimal rankings cheap enough to compute routinely. We revisit 9 publications spanning LLM evaluation, benchmark leaderboards and rank aggregation, covering 48,165 ranking instances. Simple aggregation rules such as Borda counts, win ratios and greedy cycle removal, used in five of these publications, incur 4.8–13.8% higher objective cost on average, and on 23–96% of instances select a winner that no optimal ranking places first. On a released SuperGLUE leaderboard snapshot, for instance, Borda aggregation crowns a model that provably cannot come first in any optimal ranking. XRank instead finds provably optimal rankings for 99.93% of these instances, typically within milliseconds on a single core, showing that in many settings there is no need for heuristics anymore. We release the XRank software package to facilitate optimal preference aggregation in practice.

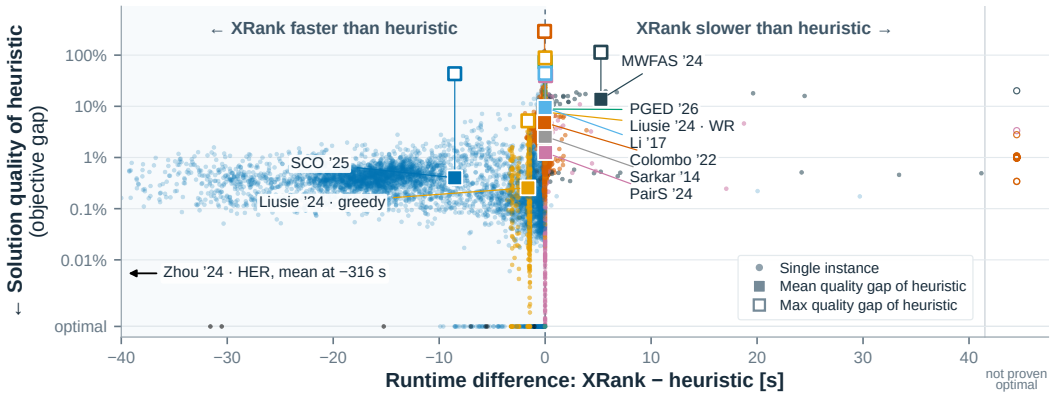


Figure 1: **Heuristics from prior work return suboptimal rankings. XRank routinely finds a proven optimum at comparable runtime.** Each square summarizes one of the ten heuristics used in the nine publications over its instances. *Y-axis:* relative objective gap $(C(\hat{\pi}) - C^*)/C^*$ between the heuristic's ranking and the optimal one (lower is better), mean (filled square) and worst case (open square). *X-axis:* Difference between XRank mean runtime and the heuristic's (left: XRank is faster). Small dots show the individual instances, colored by publication. The strip on the right shows the 25 instances XRank did not prove optimal within 5 minutes.

1 INTRODUCTION

Aggregating pairwise comparisons into a ranking is central to many machine-learning pipelines, from evaluating language models to building benchmark leaderboards and combining retrieval re-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sults (Liu et al., 2024; Liusie et al., 2024; Colombo et al., 2022; Tang et al., 2024). These comparisons are often noisy or probabilistic and contradict each other, so the aggregation step must reconcile conflicting evidence. A principled way to do so is to choose the ranking that best explains the observed comparisons, i.e., the maximum-likelihood ranking (Young, 1988).

Finding this ranking is known in combinatorial optimization as the weighted *linear ordering problem* (Martí & Reinelt, 2011), which is NP-hard (Karp, 1972; Bartholdi et al., 1989). Many publications therefore resort to simple heuristics, such as greedily discarding inconsistent comparisons or counting how many comparisons each item wins (Colombo et al., 2022; Hu et al., 2026; Liu et al., 2024; Liusie et al., 2024). The resulting errors can change the conclusions drawn from a ranking, such as which system tops a benchmark leaderboard. On the SuperGLUE snapshot released by Colombo et al. (2022), for instance, the heuristic aggregation ranks *SS-MoE* first, although provably no optimal ranking places it there. Exact methods exist (Grötschel et al., 1984; Mitchell & Borchers, 2000; Baharev et al., 2021; Grötschel et al., 2022; Stein & Andres, 2024) but are often too slow for routine use, and the few applications that aggregate exactly rely on textbook integer programs that are orders of magnitude slower than necessary (Tang et al., 2024; Korikov et al., 2025).

In this work, we revisit this problem and introduce XRank, a new exact algorithm that makes optimal rankings practical to compute for the instances arising in these pipelines. At its core is a new integer programming formulation for the linear ordering problem that operates only on the actually observed (possibly sparse) preference pairs and enforces transitivity through chordless-cycle constraints. We prove that these constraints suffice for exact ranking, analyze the associated polytope, and investigate why the formulation outperforms the classical ones from the literature. XRank combines this formulation with an efficient solver that leverages preprocessing and efficient constraint separation to quickly solve the instances to provable optimality.

We evaluate XRank on 9 publications spanning LLM evaluation, benchmark leaderboards and rank aggregation, covering 48,165 instances with up to 1,000 items. For each publication, we run its aggregation method and XRank on identical inputs, built from released or regenerated evidence, under its own objective. XRank substantially outperforms specialized exact solvers from the optimization literature, both on these instances and on established benchmark suites. The simple heuristics used in five of these publications, such as win ratios and greedy cycle removal, incur 4.8–13.8% higher objective cost on average and up to 291% on individual instances, and on 23–96% of instances select a winner that no optimal ranking places first (Figure 1). XRank removes this loss, computing a provably optimal ranking for 99.93% of the instances, often within milliseconds on a single core and thus at a running time comparable to that of the heuristics themselves. This shows that in many practical settings there is no need for heuristics anymore.

2 Background

2.1 Maximum-likelihood rankings from pairwise probabilities

Let V be a set of n items and π a ranking of V; we write $i \prec_\pi j$ when π places i before j. We encode π by one binary variable per ordered pair of distinct items,

$$x_{ij} = \begin{cases} 1 & \text{if } i \prec_\pi j, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{so that} \qquad x_{ij} + x_{ji} = 1. \tag{1}$$

Pairwise evidence. For each observed pair $\{i, j\}$, a judge (for instance an LLM comparing two responses) reports a probability $p_{ij} \in (0, 1)$ that i should precede j, with $p_{ji} = 1 - p_{ij}$. We write each observed pair once, as the arc (i, j) in its preferred direction $p_{ij} > \frac{1}{2}$. These arcs form the *comparison graph* $G = (V, A)$. Typically only a fraction of the $\binom{n}{2}$ pairs are compared. Ties ($p_{ij} = \frac{1}{2}$) contribute the same factor to every ranking and are omitted. Following Liu et al. (2024), we treat the reports as independent factors. A ranking commits, for every arc, to one of its two orders, so by equation 1 its likelihood is

$$L(x) \;=\; \prod_{(i,j)\in A} p_{ij}^{\,x_{ij}}\, p_{ji}^{\,x_{ji}} \;=\; \prod_{(i,j)\in A} p_{ij}^{\,x_{ij}}\,(1 - p_{ij})^{\,1-x_{ij}}. \tag{2}$$

Table 1: Ranking objectives in the form of equation 4.

Objective	Weight w_{ij} of arc (i, j)
Pairwise likelihood (Liu et al., 2024)	$\log\big(p_{ij}/(1-p_{ij})\big)$
Kemeny, i.e. Mallows MLE (Kemeny, 1959; Mallows, 1957)	#input rankings with $i \prec j$ minus #with $j \prec i$
Noisy comparisons (Braverman & Mossel, 2008)	net log-odds $\log\big(q_e/(1-q_e)\big)$ of observations e
Min. feedback arc set (Vahidi & Koutis, 2024)	weight of edges $i \to j$ minus weight of $j \to i$
Max. acyclic subgraph (Grötschel et al., 1985)	

Taking logarithms,

$$\log L(x) \;=\; \underbrace{\sum_{(i,j)\in A} \log(1-p_{ij})}_{\text{constant}} \;+\; \sum_{(i,j)\in A} x_{ij} \underbrace{\log \frac{p_{ij}}{1-p_{ij}}}_{=:\, w_{ij}}. \tag{3}$$

The weight w_{ij} is the log-odds of the preference, so stronger preferences weigh more. Since $x_{ij} = 1 - x_{ji}$, the maximum-likelihood ranking minimizes the total weight of *reversed* preferences:

$$\min_{\text{ranking } x} C(x) \;=\; -\log L(x) \;=\; \sum_{(i,j)\in A} w_{ij}\, x_{ji} \;+\; C_0, \qquad C_0 = -\sum_{(i,j)\in A} \log p_{ij}. \tag{4}$$

General preference weights. Nothing in equation 4 is specific to likelihoods: each arc (i, j) has a weight $w_{ij} > 0$, which a ranking pays if it places j before i. Positive weights are no restriction, because flipping an arc negates its weight and only changes the constant. Many standard ranking objectives differ only in their weights (Table 1), so from here on w_{ij} is an arbitrary positive preference weight. Problem (4) is then the weighted *linear ordering problem* (LOP).

If the preference graph contains no directed cycle, any topological order reverses no edges and is an optimal ranking. However, judges are often inconsistent in practice: a cycle $a \to b \to c \to a$ forces every ranking to reverse at least one of its arcs. Choosing the cheapest set of reversals is NP-hard (Karp, 1972; Bartholdi et al., 1989). NP-hardness rules out a polynomial-time algorithm, not exact solutions at the sizes that arise in practice. How large an instance an exact method can handle depends largely on how the problem is formulated.

3 THE PFAS FORMULATION

To solve (4) exactly, we want a problem formulation that stays small on the sparse comparison graphs that arise in practice. We propose the *Permutation-based Feedback Arc Set* (PFAS) formulation that achieves this with one decision per compared pair: keep its preferred direction, or pay to reverse it.

Select an orientation for each arc. For each arc $e = (i, j) \in A$, a binary variable x_e selects a direction. Setting $x_e = 0$ keeps i before j at no cost; setting $x_e = 1$ places j before i at cost w_e. The objective is the total reversal cost $\sum_{e\in A} w_e x_e$. We write H_x for the graph of arcs selected by x, one per compared pair. Every ranking π induces such an assignment x^π, with $x^\pi_{(i,j)} = 1$ if and only if $j \prec_\pi i$.

Consistency. Not every assignment is compatible with a ranking. A directed cycle in H_x would require an item to appear before itself, so no ranking can follow all of its arcs. Conversely, if H_x contains no directed cycle, the items can be arranged so that every selected arc points forward. Such an arrangement is a *topological order* of H_x, and it is the desired ranking. In Figure 2a, reversing (c, a) breaks the only cycle at cost 1, and the resulting directions agree with $a \prec b \prec c$. What remains is to express the requirement that H_x be acyclic with linear inequalities.

To prevent a directed cycle, at least one of its arcs must remain unselected. Since a cycle may combine preferred and reversed directions, we consider both directions of every preference pair. For each arc $e = (i, j) \in A$, define its reverse $\overleftarrow{e} = (j, i)$ and let $\overleftarrow{A} = \{\overleftarrow{e} : e \in A\}$ be the set of reverse-preference arcs. The *bidirected graph* $\overleftrightarrow{G} = (V, A \cup \overleftarrow{A})$ therefore contains both directions of every preference pair, represented by the same binary variable x_e.

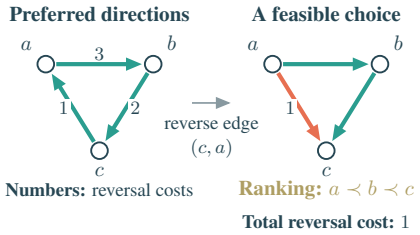


Figure 2a: **Choose directions consistent with a ranking.** Each pair either keeps its preferred direction (green) or selects the reverse arc (red), paying the indicated cost. Reversing (c, a) breaks the cycle at total cost 1.

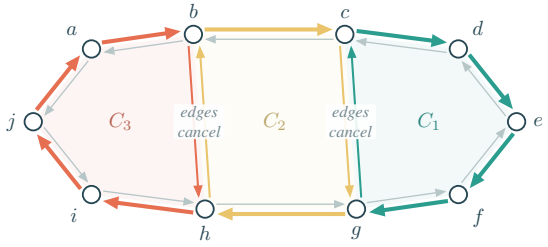


Figure 2b: **Chordless cycles suffice.** The outer cycle has chords $\{b, h\}$ and $\{c, g\}$. Each chord is used in opposite directions by two adjacent colored cycles and contributes $x_e + (1 - x_e) = 1$ to the sum of their inequalities, so summing up the three inequalities gives $\ell_C(x) + 2 \geq 3$, i.e., the outer cycle $\ell_C(x) \geq 1$.

A preferred arc e is unselected when $x_e = 1$, and its reverse $\bar{e}$ when $x_e = 0$. Thus, for a directed cycle C in $\vec{G}$,

$$\ell_C(x) := \sum_{e \in C \cap A} x_e + \sum_{\bar{e} \in C \cap \bar{A}} (1 - x_e)$$

counts its unselected arcs. Requiring $\ell_C(x) \geq 1$ prevents the cycle from appearing in H_x.

Chordless cycles suffice. A *chord* connects two nonconsecutive vertices of a cycle; a cycle without such a connection is *chordless*. In $\vec{G}$, a chord splits a cycle C into two shorter cycles C_1 and C_2 that use it in opposite directions (see Figure 2b). Summing up their inequalities counts every arc of C once, while the chord contributes $x_e + (1 - x_e) = 1$:

$$\ell_{C_1}(x) + \ell_{C_2}(x) = \ell_C(x) + 1, \qquad \text{so} \qquad \ell_{C_1}(x) \geq 1,\ \ell_{C_2}(x) \geq 1 \implies \ell_C(x) \geq 1. \tag{5}$$

The inequality of C is therefore redundant, and repeated splitting shows that chordless-cycle inequalities suffice. In Figure 2b, the three colored cycles imply the outer-cycle inequality in this way.

The PFAS optimization problem. We therefore minimize the total reversal cost while enforcing the chordless-cycle inequalities. This gives a *binary integer linear program* (ILP): an optimization problem with zero-or-one variables, a linear objective, and linear constraints, namely

$$\begin{aligned} \min_{x} \quad & \sum_{e \in A} w_e x_e \\ \text{subject to} \quad & \ell_C(x) \geq 1 \qquad C \text{ a chordless cycle in } \vec{G}, \\ & x \in \{0, 1\}^A. \end{aligned} \tag{6}$$

As derived and more formally proven in Appendix A, the formulation captures exactly the pairwise decisions that can be extended to a ranking:

Theorem 1 (PFAS is exact)**.** *The feasible solutions of (6) are exactly the assignments induced by rankings. Any topological order of an optimal solution's selected graph is an optimal ranking for (4).*

Irredundancy of the formulation. One might ask whether some chordless-cycle constraints can also be omitted. We prove that each is necessary. In fact, each defines a *facet*: a boundary face of dimension one less than the polytope itself. The following theorem is proven in Appendix A.

Theorem 2 (Facet-defining constraints)**.** *Every chordless-cycle inequality $\ell_C(x) \geq 1$ of equation 6 and every bound $x_e \geq 0$ and $x_e \leq 1$ defines a facet of the PFAS polytope, i.e., the convex hull of the integer feasible points of equation 6, and of its LP relaxation.*

4 RELATIONSHIP TO CLASSICAL FORMULATIONS

The benefit of PFAS becomes clearer when compared with two classical alternatives.

Feedback arc set: deleting preferences. FAS deletes arcs until the remaining graph is acyclic (Grötschel et al., 1985; 2022; Baharev et al., 2021). With $x_e = 1$ indicating deletion, every original directed cycle must lose at least one arc:

$$\begin{aligned} \text{(FAS)} \qquad \min_{x} \quad & \sum_{e \in A} w_e x_e \\ \text{subject to} \quad & \sum_{e \in C} x_e \geq 1 \qquad C \text{ a directed cycle in } G, \\ & x \in \{0,1\}^A. \end{aligned} \tag{7}$$

FAS and PFAS both use $|A|$ variables, but deleting a preference leaves its order undecided, whereas reversing it imposes the opposite order (Figure 3, bottom). Reversing a feasible FAS deletion set can therefore create new cycles, which PFAS excludes. An optimal FAS solution nevertheless permits recovery of an optimal ranking.

Linear ordering: ordering every pair. LO explicitly orders every pair, including those without a preference. With $y_{ij} = 1$ meaning that i precedes j, each pair chooses one direction and triangle inequalities exclude three-item cycles (Grötschel et al., 1984):

$$\begin{aligned} \text{(LO)} \qquad \min_{y} \quad & \sum_{(i,j) \in A} w_{ij}\, y_{ji} \\ \text{subject to} \quad & y_{ij} + y_{ji} = 1 \qquad i \neq j, \\ & y_{ij} + y_{jk} + y_{ki} \leq 2 \qquad i, j, k \text{ distinct}, \\ & y_{ij} \in \{0,1\} \qquad i \neq j. \end{aligned} \tag{8}$$

Because every pair is ordered, excluding these triangles rules out all directed cycles. After eliminating complementary variables, LO uses $\binom{n}{2}$ variables and $2\binom{n}{3}$ distinct triangle inequalities. PFAS avoids variables for pairs without a preference (Figure 3, right). On complete support, only triangles are chordless, and PFAS reduces to LO up to a change of variables and an additive objective constant.

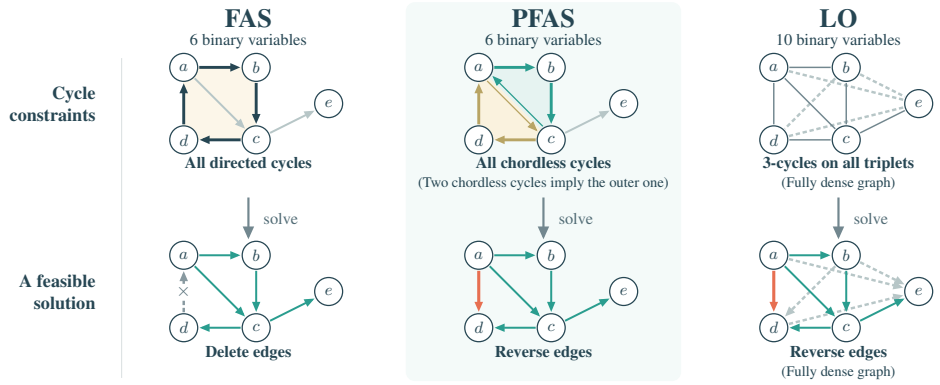


Figure 3: **Three ways to represent a ranking.** *Top:* FAS constrains all cycles; PFAS derives the outer-cycle constraint from two chordless cycles; LO operates on the complete graph and constrains 3-cycles. *Bottom:* the shown deletions and orientations are compatible with $a \prec b \prec c \prec d \prec e$.

Related formulations. Equivalent orientation-based cycle constraints have appeared in path-constrained orientation problems on undirected graphs (Figueiredo et al., 2008). However, their polyhedral analysis concerns a different polytope, which permits unoriented edges and imposes additional path constraints, so its facet results do not transfer directly to PFAS. We develop the chordless-cycle formulation for exact ranking, prove that its inequalities define facets, analyze its size relative to LO and FAS, establishing lower facet counts, and demonstrate its computational advantages over the classical formulations through matched implementation ablations.

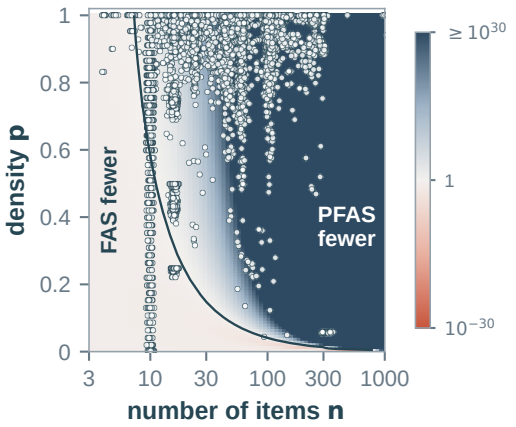


Figure 4a: **Where PFAS has fewer constraints.** Expected ratio of constraint-count between FAS and PFAS: Slate: PFAS has fewer, coral: FAS has fewer, off-white: similar counts. The black curve marks equality. Each real-world ML instance is plotted as a dot (#items, density).

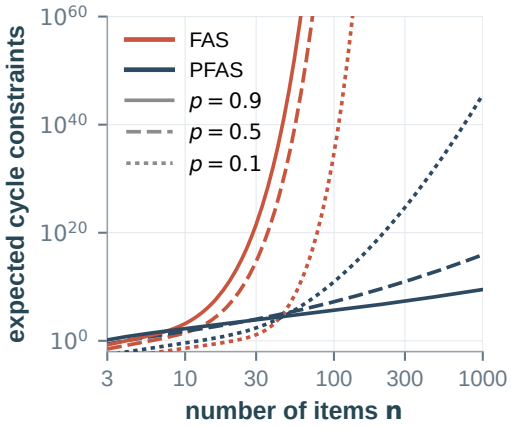


Figure 4b: **The difference grows with instance size.** Expected constraint counts of PFAS and FAS at three fixed densities w.r.t. the number of items.

Prior exact algorithms. Classical LOP cutting-plane algorithms repeatedly solve an LP relaxation and add violated inequalities (Grötschel et al., 1984). Mitchell and Borchers developed methods for solving these LP relaxations, using interior-point methods (Mitchell & Borchers, 1996) and a combination of interior-point and simplex methods (Mitchell & Borchers, 2000). For FAS, exact algorithms generate cycle constraints within branch-and-cut (Grötschel et al., 2022) or iteratively extend a subset of cycle constraints (Baharev et al., 2021). Complementary to these algorithms, partial-optimality preprocessing simplifies LOP instances before optimization (Stein & Andres, 2024).

How many variables and constraints? PFAS and FAS use $|A|$ variables, whereas LO requires $\binom{n}{2}$ even on sparse instances. The difference between PFAS and FAS lies in their cycle constraints. We analyze how their numbers vary with instance size and density.

Consider Erdős–Rényi graphs $G(n,p)$ with n items and edge probability p, orienting each edge independently and uniformly. The expected numbers of FAS and PFAS cycle constraints are:

$$\begin{aligned} F(n,p) &= \sum_{k=3}^{n} \binom{n}{k} (k-1)!(p/2)^k, \\ P(n,p) &= \sum_{k=3}^{n} \binom{n}{k} (k-1)!p^k(1-p)^{k(k-3)/2}, \end{aligned} \tag{9}$$

respectively (Appendix B). A FAS cycle requires k edges with specified directions. A PFAS cycle can use either direction, but must have no chord. The factor $(1-p)^{k(k-3)/2}$ captures the absence of all possible chords, suppressing long chordless cycles as density grows.

The resulting difference can be enormous: The expected FAS constraint count exceeds that of PFAS by a factor of over 10^{90} at $n = 100$, $p = 0.5$, and over 10^{102} at $n = 200$, $p = 0.1$. Figure 4a shows where each formulation has fewer constraints in expectation. The crossover density decreases with the number of items: PFAS has fewer expected constraints above approximately $p = 0.58$ at $n = 10$, but already above $p = 0.04$ at $n = 100$. Thus, its constraint-count advantage extends to increasingly sparse graphs as instances grow. At fixed density, the difference grows by many orders of magnitude (Figure 4b). This drastic constraint reduction does not sacrifice the strength of the LP relaxation: the LP relaxations of PFAS and FAS have the same optimal value (Proposition 7 in Appendix A.3).

To relate this model to our benchmark, we plot each instance at its size n and preference density $p = |A|/\binom{n}{2}$. Of the 48,165 instances, 94% fall where the model predicts fewer PFAS constraints. The remaining 6% have at most 16 items, and XRank solves each in under 0.2 ms. Note that these

comparisons concern expected constraint counts for random graphs with matching sizes and densities, not actual cycle counts on the instances or the number of constraints generated during solving.

5 SOLVING PFAS EFFICIENTLY

PFAS can have exponentially many chordless-cycle constraints, making it impractical to construct them all in advance. Our solver, XRank, instead starts from an initial LP that relaxes integrality and contains only a small subset of the constraints; violated constraints are then added on demand (separation). This is combined with a branching procedure, following the *branch-and-cut* framework (Grötschel et al., 1984; Padberg & Rinaldi, 1991).

Finding violated constraints. For the current solution x, assign each preferred arc e length x_e and its reverse arc length $1 - x_e$. A cycle violates its inequality exactly when its total length is less than one. Finding missing constraints therefore reduces to finding short directed cycles.

Triangles are particularly important: they are the shortest relevant cycles and the only chordless cycles on complete support. We therefore first check triangles with a dedicated efficient routine. Every directed cycle contains at least one back arc after a depth-first search (DFS) (Tarjan, 1972). We first identify these *back arcs* (B-edges), which point to a vertex still on the DFS stack. For each back arc $t \to h$, we identify all triangles containing it by intersecting the out-neighbors of h with the in-neighbors of t. Each vertex $v \in N^+(h) \cap N^-(t)$ closes a triangle $t \to h \to v \to t$, whose total length we test (Figure 5).

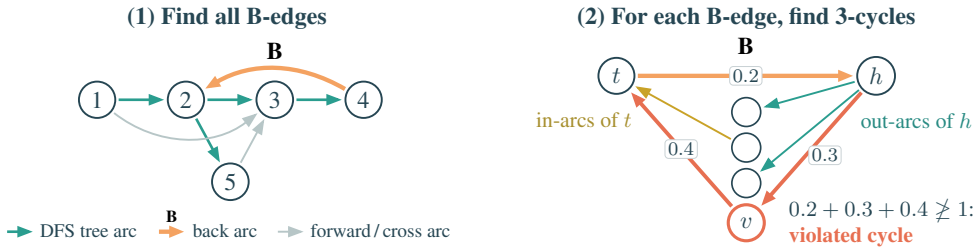


Figure 5: **Finding violated triangles.** (1) Depth-first search identifies back arcs (B-edges); every directed cycle contains one. (2) For each back arc $t \to h$, every vertex $v \in N^+(h) \cap N^-(t)$ closes a triangle. Its cycle-constraint is violated when the three arc lengths sum to less than one.

When further constraints are needed, we use a fast implementation of Dijkstra's algorithm to find longer violated cycles. For each arc $u \to v$, a shortest path from v back to u identifies the shortest cycle through that arc. This search detects a violated cycle whenever one exists. If the cycle has a chord, we split it and retain a violated shorter cycle, repeating until it is chordless. Thus, every cycle constraint we add belongs to the PFAS formulation.

Proving optimality. Our branch-and-bound procedure maintains an upper bound from the best feasible ranking found and a lower bound from the relaxations. When the bounds meet within the stated numerical tolerance, the best ranking is proven optimal. XRank is implemented in C using Gurobi 13; additional cuts, numerical tolerances, and implementation details appear in Appendix C.

6 EXPERIMENTS

6.1 SETUP

We revisit 9 publications using maximum-likelihood ranking, Kemeny aggregation, or minimum feedback arc set in their ML pipelines (Table 2). Each publication's aggregation method and XRank receive identical inputs, built from its released or regenerated evidence (Table 2), and are scored in its stated objective. All rankings are evaluated in the original objective units, including the offset C_0, and compared with those produced by the publication's own aggregation method. Reproduction

details are listed in Appendix E. All methods run on the same AMD EPYC 9745 machine using one thread with 6 GB of RAM. Details about the experimental setup can be found in Appendix C.

Table 2: Benchmark publications, and the share of instances XRank solves to proven optimality with its median time (one core, 5-minute limit). Data: (R) = evidence released by the publication, (P) = regenerated from its published protocol (Appendix E). XRank also proves optimality on all 228,469 label-ranking predictions of Zhou et al. (2024) (median below 1 ms).

						XRank	
Publication (Data)	Application	Their method	Objective	n	Instances	Optimal	Med. time
Vahidi & Koutis (2024) (R)	pairwise ranking	local ratio	FAS	19–602	80	98.8%	1.1 s
Li et al. (2017) (R)	rank aggregation	Lehmer codes	Kemeny	10–100	500	100%	2.6 ms
Hu et al. (2026) (P)	LLM response selection	greedy FAS	FAS	10–33	4,940	100%	<1 ms
Liusie et al. (2024) (P)	LLM evaluation	win ratio, greedy	likelihood	16	8,000	100%	<1 ms
Colombo et al. (2022) (R)	NLP leaderboards	Borda	Kemeny	4–1000	23,721	99.90%	6.7 ms
Sarkar et al. (2014) (P)	feature selection	approx. Kemeny	Kemeny	56–500	8	100%	3.5 ms
Liu et al. (2024) (P)	LLM evaluation	sorting	likelihood	7–300	2,028	99.95%	<1 ms
Lanctot et al. (2025) (R)	agent ranking (PrefLib)	SCO	Kemeny	11–300	8,825	99.91%	8.3 ms
Zhou et al. (2024) (R)	rank aggregation	HER	Kemeny	10–200	63	98.4%	11 ms
All					48,165	99.93%	

6.2 The cost of heuristic aggregation

For a heuristic ordering $\hat{\pi}$, we report the relative objective gap $(C(\hat{\pi}) - C^*)/C^*$ in the publication's original objective. We also test whether the heuristic's top item h can come first in *any* optimal ranking:

$$C_h^* = \min_{\pi:\, \pi_1 = h} C(\pi), \qquad \Delta_h = C_h^* - C^*. \tag{10}$$

A positive Δ_h proves that no optimal ranking places h first. Exclusion rates count only proven exclusions and retain unresolved tests in the denominator.

Table 3: **The cost of heuristic aggregation.** Optimal: fraction attaining the optimum. Winner excluded: fraction whose heuristic winner cannot appear first in any optimal ranking. Mean time: over all N inputs, using time to best for anytime heuristics and time to proven optimality for XRank (5 minutes without a proof).

						Mean time	
Heuristic	N	Mean gap	Max gap	Optimal	Winner excluded	Heuristic	XRank
MWFAS (Vahidi & Koutis, 2024)	80	13.78%	113.9%	0%	36%	701 ms	9.6 s
Lehmer codes (Li et al., 2017)	500	9.49%	44.12%	0%	96%	7.2 ms	7.4 ms
PGED (Hu et al., 2026)	4,940	8.84%	72.73%	34%	23%	<1 ms	<1 ms
Win ratio (Liusie et al., 2024)	8,000	7.70%	88.28%	1%	25%	<1 ms	1.5 ms
Borda (Colombo et al., 2022)	23,720	4.81%	291.1%	3%	27%	68 ms	305 ms
Approx. Kemeny (Sarkar et al., 2014)	8	2.59%	5.04%	25%	0%	<1 ms	7.8 ms
PairS (Liu et al., 2024)	2,028	1.25%	39.72%	53%	4%	<1 ms	187 ms
SCO (Lanctot et al., 2025)	6,662	0.40%	43.33%	28%	4%	8.6 s	86 ms
Greedy (Liusie et al., 2024)	1,600	0.26%	5.20%	43%	8%	1.6 s	<1 ms
HER (Zhou et al., 2024)	18	0.005%	0.04%	67%	0%	316 s	37 ms

Approximation loses objective and changes the winner. In five benchmark publications, the published heuristics incur 4.8–13.8% higher objective cost on average and select a winner excluded from every optimal ranking on 23–96% of instances (Table 3). Individual instances are far worse than the averages: of all 47,556 heuristic rankings, 14% lose more than 10% of the optimal objective and 685 more than 25%, with a maximum of 291%. Even a small average gap can hide large losses and winner changes: SCO and PairS have mean gaps below 1.3%, yet lose up to 43% and 40% on individual instances, and their winners are excluded on 4% of instances. Since a user cannot tell in advance whether an input is one of these, a heuristic ranking can silently distort the conclusions drawn from it; a proven optimum rules this out.

Downstream model selection. Aggregation decides which system tops a leaderboard. On the SuperGLUE snapshot released by Colombo et al. (2022), Borda selects *SS-MoE* as the winner, yet the test of Eq. (10) proves that no Kemeny-optimal ranking places SS-MoE first.

6.3 EXACT RANKING IS PRACTICAL

XRank proves optimality for 99.93% of the 48,165 instances within five minutes on one core. Median time to prove optimality is below 12 ms in every family except MWFAS (1.1 s; Table 2). Eliminating heuristic error thus often costs only milliseconds. XRank solves the integer programs of Korikov et al. (2025) in 14 ms instead of 35 s (median; Appendix E).

We compare XRank with four specialist solvers, using code provided or publicly released by their developers: GJR-LO (Grötschel et al., 1984) and GJR-FAS (Grötschel et al., 1985; 2022), based on the LO and FAS formulations, respectively; the method of Baharev et al. (2021); and S&A (Stein & Andres, 2024). We restrict the comparison to non-trivial instances (Appendix D).

XRank is fastest in every reported instance group on both the ML benchmark and classic libraries (Figure 6). It also proves optimality for every instance solved by any method in this comparison. GJR-FAS takes 10× as long on the ML benchmark and 6.6× as long on the classic libraries, using geometric means of per-instance time ratios. Results for all solvers, including solved counts, appear in Appendix D.

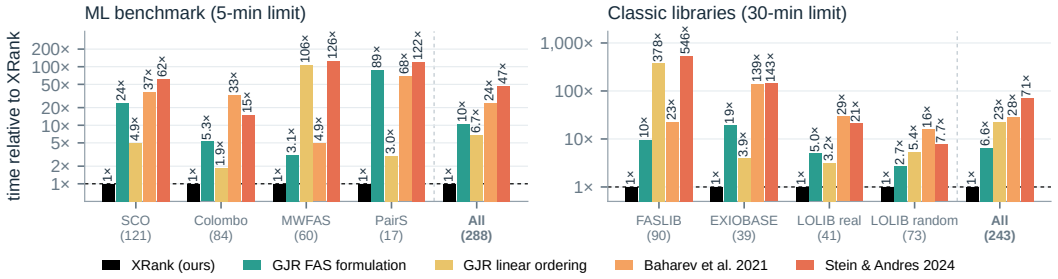


Figure 6: **XRank is faster across all reported instance groups.** Geometric-mean runtime relative to XRank (log scale) on 288 ML and 243 classic non-trivial instances solved to proven optimality by at least one method. Timeouts are charged the time limit. Parentheses note instance group sizes.

6.4 WHAT MAKES XRANK FAST: ABLATION STUDIES

Formulation. One could suspect that XRank's speed comes only from an efficient implementation. To isolate the effect of the formulation, we run the same XRank implementation, with identical preprocessing, separation and settings, on the classical FAS and LO formulations instead of PFAS. Both are slower on both benchmarks: 2.6× with FAS and 5.6× with LO on the ML benchmark, and 2.4× and 6.3× on the classic libraries, where both are slower in every density bucket (Table 6, Appendix D). LO breaks down on sparse graphs: it is 101× slower and proves only 64 of the 136 sparse instances optimal. The new formulation thus plays a large role in XRank's speed, and it is what makes XRank robust across comparison densities.

Preprocessing. Disabling all exact reductions (decomposition and DP) slows XRank only by 1.0× on the ML benchmark and 1.1× on the classic libraries (Table 6; reductions in Appendix C). Since the prior exact solvers also run without preprocessing, it explains little of the gap to them; the rest comes from the formulation and other solver components, such as separation and cut management.

7 DISCUSSION

PFAS and XRank make exact ranking practical across diverse ML applications. We measure how well each publication optimizes its own objective function; whether a better optimum also improves the downstream application depends on how well that objective models the task and is outside the scope of this paper. By making provably optimal rankings inexpensive to compute, XRank enables practitioners to eliminate heuristic optimization error from their ranking pipelines.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing and for retrieval and discovery of related work. We also used language models as experimental components: the pairwise judgments (and, for PGED, the candidate responses) of three benchmark families were regenerated with the publications' prompts and judge models, as detailed in Appendix E. We have reviewed all AI-assisted work: the authors checked every AI-suggested edit, and verified every retrieved reference against the original publication. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The PFAS formulation is stated in Section 3. Appendix A gives complete proofs of its correctness, of the facet result (Theorem 2) and of the equal LP strength of PFAS and FAS, and Appendix B states the random-graph model and derives the expected constraint counts. The solver, including preprocessing, separation, rounding, stopping rule and all parameter settings, is described in Section 5 and Appendix C. For each benchmark publication, Appendix E records the evidence, the instances, the objective and how the publication's method was run. Appendix D describes the classic benchmark suites and how the prior exact solvers were run. To facilitate optimal preference aggregation in practice, we will release the XRank software package during the review phase.

REFERENCES

Ali Baharev, Hermann Schichl, Arnold Neumaier, and Tobias Achterberg. An exact method for the minimum feedback arc set problem. *ACM J. Exp. Algorithmics*, 26:1.4:1–1.4:28, 2021.

John J. Bartholdi, III, Craig A. Tovey, and Michael A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Soc. Choice Welf.*, 6(2):157–165, 1989.

Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. In *SODA*, pp. 268–276. SIAM, 2008.

Pierre Colombo, Nathan Noiry, Ekhine Irurozki, and Stéphan Clémençon. What are the best systems? New perspectives on NLP benchmarking. In *NeurIPS*, 2022.

Fabrizio Fagiolo, Marco Baioletti, and Valentino Santucci. Linear ordering problem: Time for a change. In *PPSN (1)*, volume 16985 of *Lecture Notes in Computer Science*, pp. 399–416. Springer, 2026.

Rosa M. V. Figueiredo, Valmir Carneiro Barbosa, Nelson Maculan, and Cid C. de Souza. Acyclic orientations with path constraints. *RAIRO Oper. Res.*, 42(4):455–467, 2008.

Martin Grötschel, Michael Jünger, and Gerhard Reinelt. A cutting plane algorithm for the linear ordering problem. *Oper. Res.*, 32(6):1195–1220, 1984.

Martin Grötschel, Michael Jünger, and Gerhard Reinelt. On the acyclic subgraph polytope. *Math. Program.*, 33(1):28–42, 1985.

Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Comments on "An exact method for the minimum feedback arc set problem". *ACM J. Exp. Algorithmics*, 27:1.3:1–1.3:4, 2022.

Zhengyu Hu, Jieyu Zhang, Zhihan Xiong, Alexander Ratner, Kaize Ding, and Ranjay Krishna. Towards acyclic preference evaluation of language models via multiple evaluators. In *AAAI*, pp. 21903–21911. AAAI Press, 2026.

Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pp. 85–103. Plenum Press, New York, 1972.

John G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

Anton Korikov, Pan Du, Scott Sanner, and Navid Rekabsaz. Batched self-consistency improves LLM relevance assessment and ranking. In *EMNLP*, pp. 32687–32703. Association for Computational Linguistics, 2025.

Marc Lanctot, Kate Larson, Michael Kaisers, Quentin Berthet, Ian Gemp, Manfred Diaz, Roberto-Rafael Maura-Rivero, Yoram Bachrach, Anna Koop, and Doina Precup. Soft Condorcet optimization for ranking of general agents. In *AAMAS*, pp. 1253–1262. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

Pan Li, Arya Mazumdar, and Olgica Milenkovic. Efficient rank aggregation via Lehmer codes. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pp. 450–459. PMLR, 2017.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulić, Anna Korhonen, and Nigel Collier. Aligning with human judgement: The role of pairwise preference in large language model evaluators. In *COLM*, 2024.

Adian Liusie, Potsawee Manakul, and Mark J. F. Gales. LLM comparative assessment: Zero-shot NLG evaluation through pairwise comparisons using large language models. In *EACL (1)*, pp. 139–151. Association for Computational Linguistics, 2024.

Colin L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1/2):114–130, 1957.

Rafael Martí and Gerhard Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*, volume 175 of *Applied Mathematical Sciences*. Springer, 2011.

John E. Mitchell and Brian Borchers. Solving real-world linear ordering problems using a primal-dual interior point cutting plane method. *Ann. Oper. Res.*, 62(1):253–276, 1996.

John E. Mitchell and Brian Borchers. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In Hans Frenk, Kees Roos, Tamás Terlaky, and Shuzhong Zhang (eds.), *High Performance Optimization*, volume 33 of *Applied Optimization*, pp. 349–366. Kluwer Academic Publishers, 2000. doi: 10.1007/978-1-4757-3216-0_14.

Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.*, 33(1):60–100, 1991.

Chandrima Sarkar, Sarah Cooley, and Jaideep Srivastava. Robust feature selection technique using rank aggregation. *Appl. Artif. Intell.*, 28(3):243–257, 2014.

David Stein and Bjoern Andres. Partial optimality in the linear ordering problem. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46514–46529. PMLR / OpenReview.net, 2024.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. In *NAACL-HLT*, pp. 2327–2340. Association for Computational Linguistics, 2024.

Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

Soroush Vahidi and Ioannis Koutis. Minimum weighted feedback arc sets for ranking from pairwise comparisons. *CoRR*, abs/2412.16181, 2024.

H. Peyton Young. Condorcet's theory of voting. *American Political Science Review*, 82(4):1231–1244, 1988.

Yangming Zhou, Jin-Kao Hao, and Zhen Li. Heuristic search for rank aggregation with application to label ranking. *INFORMS J. Comput.*, 36(2):308–326, 2024.

A PROOFS

Notation. The comparison graph $G = (V, A)$ is oriented, with at most one of $(u, v), (v, u)$ in A and weights $w_e > 0$. For $e = (u, v) \in A$, let $\overline{e} = (v, u)$, $\overline{A} = \{\overline{e} : e \in A\}$, and $\vec{G} = (V, A \cup \overline{A})$. All cycles below are simple directed cycles of length at least three. A cycle is chordless if no retained pair joins nonconsecutive vertices. The two opposite arcs of one pair would only give the tautology $x_e + (1 - x_e) \geq 1$, so they are excluded from the constraint family.

For $x \in \{0,1\}^A$, let H_x have arc set $\{e \in A : x_e = 0\} \cup \{\overleftarrow{e} : e \in A,\ x_e = 1\}$. Define

$$\ell_C(x) = \sum_{e \in C \cap A} x_e + \sum_{\overleftarrow{e} \in C \cap \overleftarrow{A}} (1 - x_e).$$

A ranking π induces x^π with $x^\pi_{(u,v)} = 1$ iff $v \prec_\pi u$.

A.1 Correctness of PFAS (Theorem 1)

Lemma 3 (Chordal cycles are implied). *Every $x \in [0,1]^A$ satisfying $\ell_C(x) \geq 1$ for every chordless cycle also satisfies this inequality for every cycle of $\vec{G}$.*

Proof. Induct on cycle length. A chordless cycle is already constrained. Otherwise, let a chord join nonconsecutive vertices u, v of C, and let P^{uv}, P^{vu} be the two directed paths along C. The bidirected graph contains both chord directions, giving shorter cycles $C_1 = P^{uv} \cup \{(v,u)\}$ and $C_2 = P^{vu} \cup \{(u,v)\}$. The underlying chord variable contributes x_f to one inequality and $1 - x_f$ to the other. Thus

$$\ell_{C_1}(x) + \ell_{C_2}(x) = \ell_C(x) + 1.$$

Both terms on the left are at least one by induction, proving the claim. Equivalently, the slack $\ell_C(x) - 1$ is the sum of the two shorter-cycle slacks; repeated splitting expresses it as a sum of chordless-cycle slacks. □

Lemma 4 (Feasible points extend to rankings). *For $x \in \{0,1\}^A$, the following are equivalent: (i) x is feasible for (6); (ii) H_x is acyclic; (iii) $x = x^\pi$ for some ranking π of V.*

Proof. Each term in $\ell_C(x)$ is zero exactly when the corresponding arc belongs to H_x. Hence $\ell_C(x) = 0$ iff all arcs of C lie in H_x. Since $\ell_C(x)$ is integer, the cycle inequality fails precisely in this case. The graph H_x has no loops or pairs of opposite arcs, so any cycle in it has length at least three. By Lemma 3, (i) therefore implies (ii).

For (ii)⇒(iii), choose any topological order π of H_x. If $x_{(u,v)} = 0$ then $(u,v) \in H_x$ and $u \prec_\pi v$; otherwise $(v,u) \in H_x$ and $v \prec_\pi u$. Thus $x = x^\pi$. Finally, if $x = x^\pi$, all arcs of H_x go forward in π, so no directed cycle is selected and all inequalities hold. □

Proof of Theorem 1. Lemma 4 identifies the feasible integer vectors with acyclic orientations. The topological orders of one orientation need not be unique: two rankings induce the same vector iff they agree on every retained pair. For any such ranking, $\sum_e w_e x^\pi_e$ counts exactly the weight of the backward arcs. By (4), this is $C(\pi) - C_0$. Every ranking induces a feasible vector and every feasible vector extends to a ranking at that cost, so minimizing the two formulations is equivalent. □

Proposition 5 (PFAS is the sparse projection of LO). *Let P_{LO} be the linear-ordering polytope on all unordered item pairs, with the coordinate for each $e = (u,v) \in A$ chosen as $y_{vu} = \mathbf{1}[v \prec u]$ to match PFAS. Let $P_{\mathrm{PFAS}} = \mathrm{conv}\{x^\pi\}$ be the PFAS integer hull. Then*

$$P_{\mathrm{PFAS}} = \mathrm{proj}_A(P_{\mathrm{LO}}).$$

Proof. Every LO incidence vector is induced by a ranking π; restricting its coordinates to A gives $x^\pi \in P_{\mathrm{PFAS}}$. Conversely, every PFAS integer point is x^π for some ranking by Lemma 4, and therefore extends to the full LO incidence vector of π. Taking convex hulls gives the equality. □

A.2 Facets of the PFAS relaxation

Let $P_{\mathrm{PFAS}} = \mathrm{conv}\{x^\pi\}$ be the integer hull and let Q_{PFAS} be the polytope given by all chordless-cycle inequalities and $0 \leq x_e \leq 1$. We show that each displayed inequality defines a facet of both polytopes. This establishes irredundancy of this relaxation; it does not assert that these are all facets of P_{PFAS}.

Proposition 6. *Both P_{PFAS} and Q_{PFAS} have dimension $|A|$.*

Proof. Suppose every x^π satisfies $a^\top x = b$. For any $(u, v) \in A$, compare a ranking with u, v first with the ranking obtained by swapping them. Their vectors differ only in coordinate (u, v), forcing $a_{(u,v)} = 0$. Hence $a = 0$ and then $b = 0$, proving full dimension of P_{PFAS}. The inclusion $P_{\mathrm{PFAS}} \subseteq Q_{\mathrm{PFAS}} \subseteq \mathbb{R}^A$ proves the other assertion. □

Proof of Theorem 2. Write $\ell_C(x) = a^\top x + b_C$, so its supporting hyperplane is $a^\top x = 1 - b_C$. The tight face F is nonempty: ordering the cycle vertices consecutively along the cycle breaks exactly its closing arc. Suppose an equation $h^\top x = d$ holds throughout F. We show that it is a scalar multiple of this hyperplane equation.

Coordinates outside the cycle. Let $f = (u, v) \in A$ be an arc whose underlying unordered pair is not used by C. Since C is chordless, at least one endpoint, say v, is outside its vertex set. Starting from any tight ranking, remove v and reinsert it immediately before or immediately after u. The two resulting rankings preserve the relative order of all cycle vertices and remain tight. They differ only in coordinate f, so $h_f = 0$. The same construction applies when both endpoints are outside the cycle.

Coordinates on the cycle. For each arc g of C, order all cycle vertices in the cyclic rotation that starts at the head of g and ends at its tail. Put all other vertices after this block, in a fixed order. This ranking breaks exactly g. Because there are no chords, vectors for two such rotations, breaking g_1 and g_2, differ only in the two coordinates indexed by the corresponding arcs of A. Define $\sigma_g = 1$ if $g \in A$ and $\sigma_g = -1$ if $g \in \overleftarrow{A}$; these are the coefficients of a. Subtracting the two tight equations gives $\sigma_{g_1} h_{g_1} = \sigma_{g_2} h_{g_2}$, where h_g denotes the coefficient indexed by the corresponding arc of A (namely g if $g \in A$, and $\overleftarrow{g}$ otherwise). Thus $h = \lambda a$. Evaluating at a tight point gives $d = \lambda(1 - b_C)$. The affine hull of F is therefore exactly one hyperplane, proving that it is a facet.

Bounds. Fix $e = (u, v) \in A$ and the face $x_e = 0$, whose rankings keep u before v. For every other arc $f \in A \setminus \{e\}$, put its endpoints in an adjacent block and place the other vertices so that u remains before v in both orders of the block. This is possible because $f \neq e$: if the arcs share an endpoint, put the remaining endpoint of e on the appropriate side of the block. Swapping the block changes only x_f. Hence any equation on the face has zero coefficient on every $f \neq e$, establishing its dimension $|A| - 1$. Reverse u, v for the face $x_e = 1$.

Each of these inequalities is valid for Q_{PFAS}, and its tight face there contains the corresponding $|A| - 1$ dimensional face of P_{PFAS}. Full dimension of Q_{PFAS} then proves the facet assertion for the relaxation as well. □

A.3 PFAS AND FAS RELAXATIONS HAVE THE SAME OPTIMAL VALUE

Every feasible point of (6) is feasible for (7), and the inclusion is strict: removing every arc of a directed cycle is FAS-feasible, but reversing all of them creates the reverse cycle (Figure 3). One might therefore expect PFAS to have a tighter LP relaxation. The following proposition shows that this is not the case for the optimal value.

Proposition 7. *Let $w_e > 0$ for all $e \in A$. Every optimal solution of the LP relaxation of (7) is feasible for the LP relaxation of (6), and the latter is contained in the former. Hence both relaxations have the same optimal value.*

Proof. Containment. A directed cycle of G uses only preferred arcs, so its FAS inequality is the PFAS cycle inequality of the same cycle. If the cycle has a chord, the splitting argument (5) derives it from shorter cycle inequalities of $\vec{G}$, also for fractional $0 \leq x \leq 1$. Repeating the split shows that every FAS inequality is implied by the chordless-cycle inequalities of PFAS.

Optimal FAS points are PFAS-feasible. Let x^* be optimal for the FAS relaxation. If $x^*_e > 0$, some directed cycle $\tilde{C} \ni e$ of G is tight, $\sum_{e' \in \tilde{C}} x^*_{e'} = 1$: otherwise x^*_e could be decreased without violating any cycle inequality, and $w_e > 0$ would contradict optimality. Suppose x^* violates a PFAS inequality, $\ell_{C'}(x^*) < 1$ for a directed cycle C' of $\vec{G}$. Since x^* satisfies all FAS inequalities, C' contains a reversed arc (v, u) with $(u, v) \in A$, and $1 - x^*_{(u,v)} < 1$ gives $x^*_{(u,v)} > 0$. Take a tight cycle $\tilde{C} \ni (u, v)$. Then $\tilde{C} \setminus \{(u, v)\}$ is a directed path from v to u in G of total value $1 - x^*_{(u,v)}$. Replacing (v, u) by this path leaves the left-hand side unchanged. After replacing every reversed

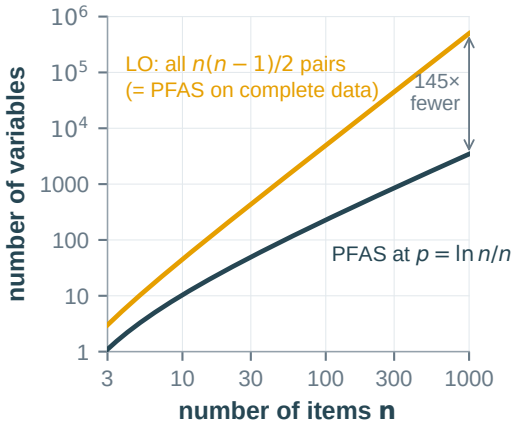


Figure 7: **LO has too many variables.** LO needs one variable per item pair, PFAS one per compared pair; on sparse data ($p = \ln n/n$) that is $n/\ln n$ times fewer.

arc, C' becomes a closed walk in G of value below 1. Since $x^* \geq 0$, this walk contains a directed cycle of G of value below 1, contradicting FAS feasibility.

Equal values. By containment, the PFAS relaxation value is at least the FAS value, and the PFAS-feasible optimal FAS point gives the reverse inequality. □

The advantage of PFAS is therefore not a stronger bound at the root. It lies in the structure of the constraint system: far fewer inequalities in the dense regime (Figure 4), each of them facet-defining (Theorem 2).

B Expected constraint counts

The undirected support follows the Erdős–Rényi model $G(n, p)$: retain each of the $\binom{n}{2}$ unordered pairs independently with probability p and orient each retained pair independently in either direction with probability $1/2$. The resulting graph is oriented, with no tied comparisons. Thus a specified arc of G is present with probability $p/2$, while a specified edge of the underlying undirected support graph is present with probability p. Retaining that edge puts both corresponding arcs in $\vec{G}$. These assumptions describe the random model, not the benchmark's preference-generation processes.

FAS. A set of k vertices has $(k-1)!$ directed cyclic orders, counting rotations as identical and opposite directions as distinct. Each occurs in G with probability $(p/2)^k$. Summing indicator expectations gives

$$\mathbb{E}[\#\text{FAS cycle constraints}] = \sum_{k=3}^{n} \binom{n}{k} (k-1)!(p/2)^k = F(n,p).$$

PFAS. A directed cyclic order is present in $\vec{G}$ when its k undirected support edges are retained. It is chordless precisely when the other $\binom{k}{2} - k = k(k-3)/2$ pairs among those vertices are absent. These events concern distinct unordered pairs and are independent, giving

$$\mathbb{E}[\#\text{PFAS cycle constraints}] = \sum_{k=3}^{n} \binom{n}{k} (k-1)!p^k(1-p)^{k(k-3)/2} = P(n,p).$$

Cycles may overlap; independence between cycle indicators is unnecessary for these expectations. For $p = 1$, only the $k = 3$ term remains, with the exponent-zero factor interpreted as one.

LO and variable counts. LO has $2\binom{n}{3}$ distinct triangle inequalities and $\binom{n}{2}$ variables. Both sparse formulations have expected variable count $p\binom{n}{2}$. The cycle counts above exclude variable bounds. Figure 7 compares the variable counts.

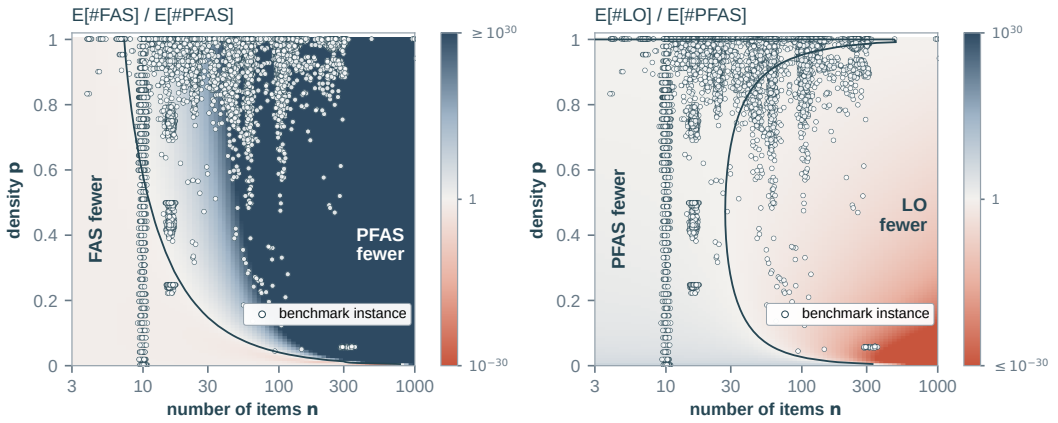


Figure 8: Expected number of FAS (left) and LO (right) relative to PFAS cycle constraints on random graphs; slate: PFAS needs fewer, off-white: about equal, coral: the other formulation needs fewer; black line: equal. Dots: benchmark instances.

For fixed $0 < p < 1$, the chord-absence factor decays as $\exp(-\Theta(k^2))$, suppressing long PFAS cycles in this model. The ratios in Figures 4 and 8 are ratios of expected counts, not expected ratios or estimates of an individual graph's counts. In particular, the orientations of the support edges matter to FAS: a transitive complete graph has zero directed cycles, whereas its PFAS formulation still has $2\binom{n}{3}$ triangle inequalities. Neither full-family count is the number of cuts generated by a separation-based solver.

C Implementation and Evaluation Details

Solver and settings. XRank is our branch-and-cut implementation of PFAS in C on the Gurobi 13 callable library. It runs a root cutting-plane loop that separates chordless cycles, followed by Gurobi's branch-and-bound with lazy cycle constraints. The instance is first split into its strongly connected components, which are solved independently; components with at most 10 items are solved directly by a subset dynamic program. All reported runs use one thread and one physical core of an AMD EPYC 9745 (6 GB memory per run, one scheduler job per instance and method). The limits are 5 minutes (ML benchmark) and 30 minutes (classic libraries). A run proves optimality when the primal and dual bounds differ by less than $1 - 10^{-4}$ for integral weights, or by less than 10^{-6} otherwise; the baselines were patched to use the same rule (Appendix D). The ablations use the same XRank source, with the FAS or LO formulation (for LO, every component is completed with zero-weight pairs) or with every exact reduction turned off; all other settings and limits are identical.

Preprocessing. All reductions are exact. Opposite arcs of a pair are netted, and the smaller cost enters the constant C_0. The graph is then split into strongly connected components: arcs between components point forward in some optimal ranking, and components are ranked independently in topological order. Decomposition repeats on each component until nothing changes. Components with at most ten items are solved by a dynamic program over subsets.

Separation. Given an LP point x, the separation graph contains, for every pair $e = (u, v) \in A$, the arc $u \to v$ with length x_e and the arc $v \to u$ with length $1 - x_e$; a PFAS inequality is violated exactly when a directed cycle has length below one. Separation runs as a cascade of increasingly expensive routines. Violated triangles come first (Figure 5): a depth-first search marks its back arcs, whose heads are still on the search stack. Since every directed cycle contains a back arc, it suffices to check, for each back arc $t \to h$, the items in $N^+(h) \cap N^-(t)$. If few triangles are found, a depth-first search on the arcs of length zero finds cycles of the current integral part, and then, for each arc $u \to v$, a Dijkstra search for the shortest path from v back to u finds the most violated cycle through that arc. The last step is exact: it returns a violated cycle whenever one exists. Every cycle found is shortcut along chords until it is chordless, which keeps only the facet-defining inequalities. When

no cycle is violated, $\{0, \frac{1}{2}\}$-Chvátal–Gomory cuts are derived at half-integral points. New rows are selected greedily by efficacy and objective parallelism, skipping rows nearly parallel to one already selected, and rows that stay non-binding are removed.

Primal solutions and branch-and-bound. Every LP solution is rounded to a ranking by a greedy topological order on the arc lengths. If the root loop leaves a gap, Gurobi's branch-and-bound continues from the root LP: the same cascade runs on fractional node solutions, and every integral solution is checked exactly for directed cycles, whose inequalities are added as lazy constraints.

Objectives and the winner test. Reported costs and bounds include the constant C_0 removed by the reduction. Returned permutations are rescored under the original objective. The winner test adds $M = 1 + \sum_{e \in A} w_e$ to the weight of every pair that places another item before h. Every ranking with h first keeps its original reversal cost, which is below M, and every other ranking costs more than M, so every optimum of the modified instance puts h first and has value C_h^*. It is solved by the same solver, and h counts as excluded only if this solve proves optimality and C_h^* exceeds C^* by more than the relative tolerance 10^{-7}. It runs separately from the ranking solve, and its time is not part of any reported ranking time.

D Classic LOP/FAS benchmarks

Implementations. All four baselines are the authors' implementations, none a reimplementation: the two GJR codes were provided by their authors, and the codes of Baharev et al. (2021) and Stein & Andres (2024) are their public GitHub releases. *GJR-LO* is the linear-ordering cutting-plane code in the line of Grötschel et al. (1984): all $n(n-1)/2$ pair variables, separated 3-cycle inequalities, and Gurobi for the integer phase. *GJR-FAS* is the FAS code of Grötschel et al. (2022): one variable per arc, Dijkstra-based cycle separation at fractional points in a root cutting-plane loop, LP rounding, then Gurobi branch-and-cut with lazy cycle cuts. Baharev et al. (2021) solve the set-cover formulation over directed cycles, generating cycles lazily from incumbents inside Gurobi's branch-and-cut (their `sdopt-tearing` code, run through the entry point of their benchmarks). Stein & Andres (2024) decompose the instance with partial-optimality conditions and solve the remaining subproblems as LO integer programs with Gurobi (their released C++ code, subcommand `pi`). *Modifications.* We changed the baseline codes only where needed for a common evaluation protocol, never their algorithms. Every solver uses one thread and a wall-clock limit. All use the stopping rule of Appendix C; GJR-FAS, Baharev et al. and S&A originally stopped at Gurobi's default relative gap of 10^{-4}, which does not prove optimality. Every solver reports its best ordering and its dual bound on a timeout, and each returned ordering is rescored under the original objective. In addition, two small bug fixes let the GJR codes run on all instances. These modifications will be part of our code release.

Instances. All 247 FASLIB instances, the 203 LOLIB instances of the IO, SGB, Spec, RandA2 and RandB classes that the LOLIB documentation lists as solved, and the 175 EXIOBASE input–output tables of Fagiolo et al. (2026), 625 in total. The random classes RandA2 and RandB are marked in Table 4. The ML benchmark consists of the instances of Table 2.

Non-trivial instances. The comparison with the specialist solvers (Figure 6, Tables 4 and 5) uses the instances on which XRank in its PFAS, FAS or LO variant (Section 6.4) needs at least 1 s or does not prove optimality, and which at least one method solves to proven optimality. Including the tens of thousands of instances that all three variants solve within 1 s would reduce the comparison to millisecond timings. The selection removes only instances that are easy for XRank, so, if anything, it works against XRank. Runs without an optimality proof are charged the time limit, and per-instance time ratios use a 1 ms floor.

Results by family. Table 4 breaks the comparison of Section 6 down by instance family.

Ablation. Table 6 gives the matched-solver ablation of Section 6.4 by benchmark and density regime, on the instances of Table 5, and Figure 9 shows the formulation ablation on the classic libraries by comparison density.

Table 4: Exact solvers per instance family, on the non-trivial instances (XRank in its PFAS, FAS or LO variant needs at least 1 s or does not prove optimality) that at least one method proves optimal; one CPU core. XRank: instances proven optimal / shifted geometric mean time in seconds (shift 1 s). Other columns: instances proven optimal / slowdown, the geometric mean of the per-instance time ratio to XRank (times floored at 1 ms; runs without an optimality proof charged the limit, which makes those ratios lower bounds). *random instances; †partly synthetic.

Instances	n	XRank	GJR-FAS	GJR-LO	Baharev	S&A
ML benchmark (5-min limit)						
SCO '25	121	**121 / 3.43**	97 / 24×	105 / 4.9×	74 / 37×	65 / 62×
Colombo '22	84	**84 / 0.359**	83 / 5.3×	84 / 1.9×	83 / 33×	83 / 15×
MWFAS '24	60	**60 / 3.03**	59 / 3.1×	15 / 106×	58 / 4.9×	0 / 126×
other (3 groups)	23	**23 / 4.11**	16 / 36×	19 / 2.8×	13 / 42×	8 / 54×
all	288	**288 / 2.11**	255 / 10×	223 / 6.7×	228 / 24×	156 / 47×
Classic libraries (30-min limit)						
FASLIB†	90	**90 / 4.83**	72 / 10×	26 / 378×	66 / 23×	8 / 546×
LOLIB RandB*	50	**50 / 44.8**	44 / 2.1×	33 / 4.3×	33 / 8.3×	38 / 4.6×
EXIOBASE	39	**39 / 1.82**	39 / 19×	39 / 3.9×	38 / 139×	35 / 143×
LOLIB SGB	25	**25 / 1.93**	25 / 5.6×	25 / 1.7×	25 / 56×	25 / 18×
LOLIB RandA2*	23	**23 / 11.8**	23 / 4.8×	20 / 9.1×	14 / 67×	21 / 24×
LOLIB Spec†	16	**16 / 3.74**	16 / 4.2×	16 / 8.6×	16 / 11×	14 / 26×
all real / mixed	170	**170 / 3.37**	152 / 10×	106 / 42×	145 / 36×	82 / 183×
all	243	**243 / 6.85**	219 / 6.6×	159 / 23×	192 / 28×	141 / 71×

Table 5: **Comparison with specialist exact solvers.** Non-trivial instances solved to proven optimality by at least one method. Slowdown: geometric mean of per-instance time ratios relative to XRank. SGM: shifted geometric-mean runtime.

	ML benchmark (288 inst., 5 min)			Classic libraries (243 inst., 30 min)		
Method	Optimal	SGM [s]	Slowdown	Optimal	SGM [s]	Slowdown
XRank (ours)	**288**	**2.11**	**1**×	**243**	**6.85**	**1**×
GJR-FAS (Grötschel et al., 2022)	255	15.2	10×	219	37.7	6.6×
GJR-LO (Grötschel et al., 1984)	223	12.7	6.7×	159	112	23×
Baharev et al. (Baharev et al., 2021)	228	23.3	24×	192	141	28×
S&A (Stein & Andres, 2024)	156	55.8	47×	141	322	71×

E PER-PUBLICATION BENCHMARK DETAILS

Per-publication records. For every row of Table 2: the evidence (R = released, P = regenerated), the instances, the objective, and how the publication's method was run.

Zhou et al. (2024) (R). Authors' code and data. The 63 rank-aggregation instances ($n = 10$–200) are the nine real-world profiles, the same files read as item lists, and 45 truncated incomplete

Table 6: Ablation on the instances of Table 5: the same XRank implementation with the FAS or LO formulation instead of PFAS, or with every exact reduction turned off. Slowdown relative to XRank as in Table 5; density c is the fraction of item pairs compared.

	ML (288)		Classic (243)		Sparse ($p < 0.3$) (136)		Dense ($p \geq 0.9$) (238)	
Formulation	Optimal	Slowdown	Optimal	Slowdown	Optimal	Slowdown	Optimal	Slowdown
XRank (PFAS)	**288**	**1**×	**243**	**1**×	**136**	**1**×	**238**	**1**×
FAS formulation	286	2.6×	235	2.4×	135	1.5×	232	3.3×
LO formulation	238	5.6×	216	6.3×	64	101×	234	1.9×
No preprocessing	287	1.0×	242	1.1×	135	1.2×	237	1.0×

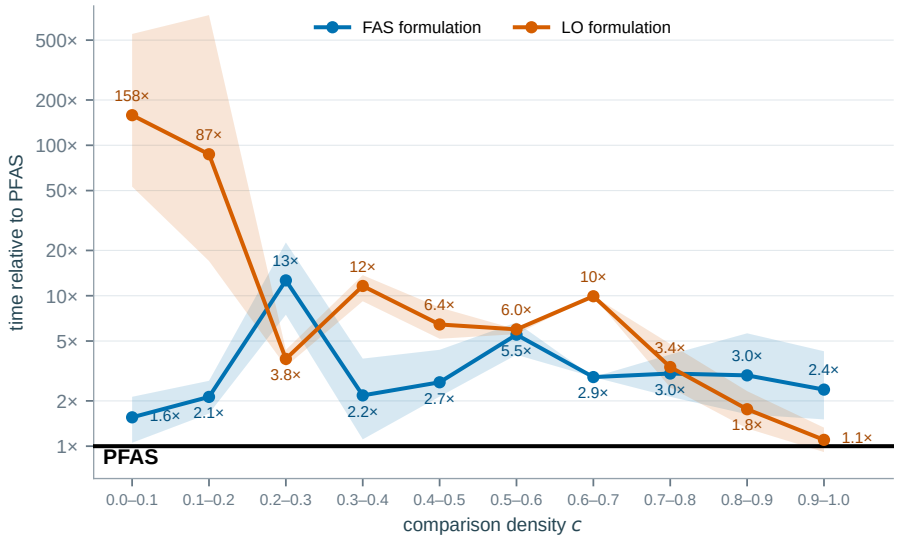


Figure 9: **Only PFAS is fast at every density.** The same XRank implementation with the FAS or LO formulation, relative to PFAS, on the classic libraries. Lines: geometric mean; shaded: middle 50% of instances. Buckets between 0.2 and 0.8 contain few instances.

profiles; the synthetic Mallows profiles are excluded. Objective: total Kendall disagreement with the input rankings. Baseline: the authors' `HER.cpp` with the publication's settings; it has no objective for incomplete profiles, which are solved exactly only. The label-ranking instances come from the authors' Label Ranking Forest setup.

Sarkar et al. (2014) (P). No code released. Instances rank the features of the publication's public datasets ($n \leq 1000$: Lung Cancer, mfeat-fourier, Arrhythmia, Madelon) by the publication's three Weka rankers, regenerated at default options; ties are kept as ranked or treated as no preference. Objective: Kemeny disagreement. Baseline: our implementation of median-rank aggregation, standing in for the unspecified Kemeny approximation.

Hu et al. (2026) (P). Authors' code, which ships no judgments. The 4,940 instances ($n = 10$–33) are regenerated with their prompts and the publication's small evaluators on HumanEval, GSM8K and MATH, plus AlpacaEval model rankings. Objective: the removed feedback-arc weight of the evaluator ensemble. Baseline: their Algorithm 1, which removes the same weight as the released greedy code on every instance.

Liu et al. (2024) (P). Authors' code; the probability tables were not released and were regenerated with PairS' prompts, both presentation orders and open-weight judges. The 2,028 instances are NewsRoom ($n = 7$), SummEval ($n = 16$) and dense HANNA subsets ($n = 100$–300). Objective: the negative log of the pairwise likelihood. Baseline: a port of the released PairS greedy, calibrated as in the publication, from ten seeded starts.

Lanctot et al. (2025) (R). PrefLib, restricted as in the publication, which reproduces its per-bin profile counts exactly; the 8,825 instances are all profiles with 11–300 alternatives. Objective: SCO's discrete loss, Kendall disagreements over the pairs each vote ranks. Baseline: the released OpenSpiel SCO with the publication's hyper-parameters and three seeds, run for profiles with up to 200 alternatives.

Liusie et al. (2024) (P). Authors' code on SummEval; the probabilities were regenerated with the publication's judge, Flan-T5-3B, which reproduces their positional bias and win-ratio correlations. The 8,000 instances ($n = 16$) are full comparison tables and comparison subsets. Objective: the product likelihood of their Eq. (12). Baselines: the released win ratio, and the released greedy search on full tables.

Colombo et al. (2022) (R). Authors' code and leaderboard data (GLUE, SuperGLUE, XTREME and instance-level score tables). The 23,721 instances ($n = 4$–1000) are the publication's task- and instance-level aggregations and the robustness experiments of their notebooks. Objective: the Kemeny consensus. Baseline: their Borda code, unchanged, which reproduces the publication's Borda rankings and counts.

Li et al. (2017) (R). No code released; data are Jester and Sushi. The 500 instances are voter samples from the complete users ($n = 100$ and $n = 10$); the synthetic Mallows profiles are excluded. Objective: total Kendall distance. Baseline: our implementation of their Lehmer-code aggregation (Algorithm 2).

Vahidi & Koutis (2024) (R). Authors' code and graphs; the 80 graphs ($n = 19$–602) are those of the publication's tables plus three further released faculty-hiring graphs. Objective: the weighted feedback arc set. Baseline: the released code, run unchanged, which reproduces the publication's reported losses.

Exact aggregation of Korikov et al. (2025) (Section 6). Their code and LLM outputs are not available, so we regenerated the listwise rankings with their prompt and sampling settings on BM25 candidate lists for TREC DL-19 and TREC-COVID, following their protocol, using Qwen3.8-27B in place of their commercial judges. Each of the 2,790 instances aggregates 1–15 sampled rankings of a query's top 90 passages under the Kemeny objective. Their aggregator is the exact Kemeny program of Tang et al. (2024): the LO formulation (8) with all pair variables and all triangle inequalities, built in PuLP and solved by CBC. We ran this released code unchanged on the same machine and core as XRank and timed model construction and solve. Its median is 35 s, with 6 instances unfinished after 30 minutes and counted at that limit. XRank proves optimality on all 2,790 instances with a median of 14 ms, and both return the same optimum wherever CBC finishes.

Results and accounting. Table 2 reports, per family, the number of instances, the share XRank solves to proven optimality and the median time to prove it. The 34 runs without an optimality proof end with a mean remaining gap of 27.0% on the 23 MLQE leaderboards, 0.35% on the 8 PrefLib profiles, and 0.46% (PairS), 0.67% (Zhou et al., 2024) and 10.5% (MWFAS) on the single others. Table 3 counts fewer inputs than Table 2 where a heuristic does not apply: the released HER code has no working objective for the 45 incomplete profiles of Zhou et al. (2024), the released greedy search of Liusie et al. (2024) cannot take comparison subsets (6,400 inputs), SCO is run only up to 200 items (2,083 PrefLib profiles), and the 81 inputs with optimum 0, where the relative gap is undefined, are omitted. The runs without an optimality proof that have a heuristic are 25 of the 34; the others are PrefLib profiles above 200 items and one incomplete profile. Winner-exclusion rates use the same N inputs, and an unresolved winner test counts as not excluded.