Generalized Priority-Aware Shapley Value

Kiljae Lee
The Ohio State University
lee.10428@osu.edu

Ziqi Liu
Carnegie Mellon University
ziqiliu2@andrew.cmu.edu

Weijing Tang
Carnegie Mellon University
weijingt@andrew.cmu.edu

Yuan Zhang*
The Ohio State University
yzhanghf@stat.osu.edu

Abstract

Shapley value and its priority-aware extensions are widely used for valuation in machine learning, but existing methods require pairwise priority to be binary and acyclic, a restriction spectacularly violated in real-data examples such as aggregated human preferences and multi-criterion comparisons. We introduce the *generalized priority-aware Shapley value* (GPASV), a random order value defined on arbitrary directed weighted priority graphs, in which pairwise edges penalize rather than forbid order violations. GPASV covers a range of classical models as boundary cases. We establish GPASV through an axiomatic characterization, develop the associated computational methods, and introduce a priority sweeping diagnostic extending PASV's. We apply GPASV to LLM ensemble valuation on the cyclic Chatbot Arena preference graph, illustrating that **priority-aware valuation is not a one-button operation**: different balances of pairwise graph priority versus individual soft priority produce substantively different valuations of the same data.

1 Introduction

Shapley value [38] and its variants are widely used in machine learning to accredit content contributors, including training examples, data providers, features, or model components [16, 32, 40]. Their appeal lies in being both principled (uniquely determined by a small set of axioms) and model-agnostic. A central axiom is symmetry: two players who contribute identical contents should receive the same credit. While this is reasonable on its face, its key vulnerability is the **copier attack**: symmetry rewards copiers equally to originators, when rationality says all credit should go to the originators. Modern AI/ML problems contain many examples where for similar or related reasons, contributors should be valued differently for reasons beyond their contributed contents: data lineage, causal precedence among features, differing trust or cost across data providers. Recent work has incorporated this contextual information into Shapley-type valuations [42, 9], including precedence Shapley values (PSV) [11], weighted Shapley values (WSV) [20, 35], machine-learning adaptations [13, 47], and most recently the priority-aware Shapley value (PASV) [26].

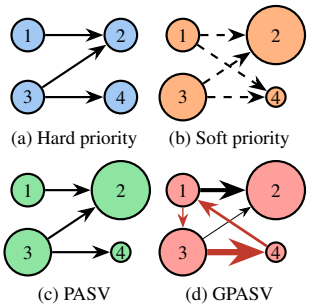


Figure 1: Illustration of priority structures. Dashed arrows in (b): [35, 47] require special precedence structures.

A common assumption across all these works is that the pairwise priority relations can be captured by a **directed acyclic graph (DAG)**. The meaning of DAG assumption is two-fold: (i) it has **no cycles** (e.g., $i \to j \to k \to i$); (ii) precedences are **unweighted**; both aspects are violated in many

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

modern AI/ML applications. A prominent example of cyclic priority is aggregated human preferences, where *Condorcet cycles* are ubiquitous [8, 2, 12, 29]; multi-criterion model comparison is another example in similar spirits [45]. Weighted priority arises whenever the strength of evidence varies, encompassing pairwise win counts in tournament analysis, Bradley-Terry log-odds [3], ELO ratings [17], and Plackett-Luce-style worth scores [36, 31]; all of these carry magnitude that a DAG discards. Some applications, such as Chatbot Arena [46, 7], present cyclic, weighted priority graphs.

We propose **generalized priority-aware Shapley value (GPASV)**, a valuation method that simultaneously handles cyclic and weighted priority graphs. Technically speaking, Shapley-style valuation methods all build on permutations of the players (see Section 2). Within this language, the DAG assumption amounts to rigidly restricting to admissible permutations. Our GPASV instead, for each permutation, counts its total amount of precedence violations and inversely weights its probability accordingly, thereby naturally handling cycles and weighted pairwise priorities.

Apart from pairwise priority, represented by the priority graph, GPASV also incorporates the player-level "*soft priority*", which encodes individual information such as trust, cost, or compliance risk. While the idea of this component was inherited from the WSV-to-PASV line of prior work, integrating it with our "upgraded" priority-graph component requires rather nontrivial adaptation.

1.1 Our Contributions

Method (core contribution): GPASV handles cyclic and weighted priority graphs while incorporating individual soft priority. GPASV is the first valuation method that simultaneously handles cyclic pairwise priorities, weighted pairwise priorities, and individual soft priorities, while still containing the classical Shapley value, PSV, WSV, and PASV as special cases.

Theory: a principled, expressive extension of PASV. GPASV admits an axiomatic characterization that generalizes PASV's, with GSCF fixing the canonical stage-wise form (Section 3.4). Furthermore, on cyclic graphs GPASV admits limiting distributions that no PASV can express (Section 3.6), marking GPASV as a highly nontrivial extension rather than a wrapper around PASV.

Scalable computation and acceleration methods. We derive a GPASV-specific local adjacent-swap MH ratio (Proposition 1) and design a stage-wise greedy initialization for cyclic graphs (Algorithm 2), then integrate these sampling components with direct Monte Carlo estimation, utility caching (Section F.3.1), SNIS reuse for priority sweeping (Section F.3.2), and a surrogate-assisted estimator adapted from regression-based semi-value methods (Section F.2.2). While some acceleration components build on standard Monte Carlo and regression-surrogate ideas, their adaptation to the GPASV permutation distribution and priority-sweeping workflow significantly enhances scalability.

Diagnostic: priority sweeping with a new hard-priority axis. We extend PASV's priority sweeping diagnostic to GPASV. Beyond PASV's soft priority sweep, GPASV's weighted graph admits a new hard priority sweep over β. Since the balance between the two priority sources is the evaluator's preference rather than a tuning parameter, GPASV reports a family of valuations along the sweep together with sensitivity diagnostics (Section 3.5).

Validation and application. Extensive simulations validate GPASV's accuracy and confirm our theoretical predictions. A large-scale experiment on LLM ensemble valuation on MT-Bench and the cyclic Chatbot Arena preference graph demonstrates GPASV's practicality and scalability.

2 Preliminaries

2.1 From Shapley Value to Random Order Values (ROV)

Shapley value (SV). Let $[n] = \{1, \dots, n\}$ be a set of players. For any $S \subseteq [n]$, let $U(S)$ be the revenue earned by the joint work of S. We call U a *utility function* with $U(\emptyset) = 0$. The Shapley value [38]

$$\nu_i(U) := \sum_{S \subseteq [n] \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} \Big\{ U(S \cup \{i\}) - U(S) \Big\}. \tag{1}$$

is a payoff method that uniquely satisfies four widely-desired axioms (see Appendix B).

Random order values (ROV). Shapley value is prone to **copier attack** – an attacker i can unfairly split j's pay by simply copying her. To defend against this attack, it is essential to consider the **order/rank** among players, thus if i consulted j's work, then j should always be present in the prefix S in (1) when evaluating i's contribution. This leads to a generic notion: **random order value (ROV)**.

Definition 1 (**Random Order Value (ROV)**). *Let $\pi = (\pi_1, \ldots, \pi_n)$ be a permutation of $[n]$ and π^i be the set of players located before i in π. The random order value (ROV) of i is defined as*

$$\nu_i^p(U) := \mathbb{E}_{\pi \sim p}\big[U(\pi^i \cup \{i\}) - U(\pi^i)\big], \tag{2}$$

where p is a distribution over Π, the set of all permutations of $[n]$.

Different choices of p yield different priority-aware extensions of Shapley value. Examples include: (i) Shapley value, with $p = \text{Uniform}(\Pi)$; (ii) precedence Shapley value (PSV) [11], with p uniform over a set of "permitted permutations" induced by a directed acyclic graph (DAG) encoding hard, pairwise priority; (iii) weighted Shapley value (WSV) [20, 35] incorporates individual-level weights to encode soft priority. Going forward, we will describe each method by its p, without repeating (2).

2.2 Priority-Aware Shapley Value (PASV)

Hard vs. soft priority. Priority information available to the user typically comes in two forms.

Hard priority specifies pairwise constraints that must be respected at all times, e.g., a causal ancestor must precede its descendant. Hard priority is encoded by a directed acyclic graph (DAG) $G = ([n], \mathcal{E})$, where an edge $(i, j) \in \mathcal{E}$ means that i must appear before j in any π.

Soft priority, in contrast, expresses individual-level preferences without forbidding any order; instead, it adjusts the ordering within the boundary permitted by the hard priority. Players who are more trusted, less costly, or carry less legal/compliance risk than others should be rewarded by an earlier spot in π, thus face less peer competition when evaluated. Soft priority is encoded by individual weights on each player: $\lambda = (\lambda_1, \ldots, \lambda_n)$, where $\lambda_i > 0$.

Priority-aware Shapley value (PASV). [26] proposed PASV to combine hard and soft priority in one ROV. For $\pi \in \Pi$ and prefix $S_t := \{\pi_1, \ldots, \pi_t\}$, let $\max(S_t)$ denote the set of players $i \in S_t$ such that no $j \in S_t$ is a descendant of i in G; these are the elements admissible to be π_t. PASV is

$$p^{(\lambda, G)}(\pi) \propto \mathbb{1}_{[\pi \in \Pi^G]} \cdot \prod_{t=1}^{n} \frac{\lambda_{\pi_t}\, |\max(S_t)|}{\sum_{k \in \max(S_t)} \lambda_k}, \tag{3}$$

where $\Pi^G := \{\pi \in \Pi : i \text{ appears before } j \text{ in } \pi \text{ for all } (i, j) \in \mathcal{E}\}$ is the set of permutations consistent with G. In (3), the indicator $\mathbb{1}_{[\pi \in \Pi^G]}$ enforces hard priority. The second part of (3) uses a *stage-wise* formulation to encode soft priority λ. This part can be roughly understood as follows: starting from position $t = n$ down to $t = 1$, π_t is drawn from the admissible set $\max(S_t)$ with probability proportional to λ_{π_t}; the resulting π's probability is then rescaled by the factor $\prod_{t=1}^{n} |\max(S_t)|$ — this rescaling is essential for PASV to satisfy its axioms (see Remark 3.3 in [26]). PASV recovers SV, PSV and WSV as special cases under proper choices of λ and G.

3 Our Method

3.1 Limitations of PASV and Motivation for a Generalization

PASV encodes hard priority as a binary DAG, which can be restrictive in modern AI/ML applications.

Hard priority may not be that "hard" in practice. In (3), every edge of G is enforced as an absolute constraint: any π violating even one edge is ruled out. But real pairwise priority is often a matter of degree rather than black/white. For example, NeurIPS 2026 draws a line at March 1, 2026: papers appearing before are treated as prior work, those after as concurrent [34], even though two papers posted one day before/after carry nearly identical priority. Similarly, a domain expert may believe feature i causally impacts feature j based on informative but non-determining evidence. In both cases, PASV discards magnitude information and becomes sensitive to borderline edges.

Pairwise priority may be cyclic. PASV requires G to be acyclic. But cyclic pairwise signals arise naturally when priority is aggregated across a population or across criteria. In aggregated human preferences, model i may beat j, j beats k, and yet k beats i – the classical Condorcet cycle [2, 12], shown to occur with high probability in LLM preference networks [29]. Multi-criterion comparison can be similarly cyclic: A beats B on criterion 1, B beats C on criterion 2, C beats A on criterion 3. Forcing such data into a DAG requires unfairly favoring certain players or biasing criteria.

3.2 Generalized Priority-Aware Shapley Value (GPASV)

To address the two limitations of PASV in Section 3.1, we first replace the binary DAG G with a weighted directed graph $(\omega_{ij})_{i,j\in[n],i\neq j}$, where each $\omega_{ij} \geq 0$ encodes the strength of the pairwise priority "i should precede j": $\omega_{ij} = 0$ means no such priority; we set $\omega_{ii} \equiv 0$ throughout. This convention directly accommodates common forms of user data – pairwise win counts, Bradley-Terry log-odds, ELO differences, or expert-assigned confidence scores – without ad-hoc transformations. Note that ω_{ij} and ω_{ji} are independent: in encoding a DAG-style precedence "$i \prec j$" at most one of them is nonzero, while in encoding pairwise comparison records (e.g., i and j's head-to-head wins), both can be positive. Unlike PASV, we impose no acyclicity requirement on ω. In practice, we recommend rescaling the input edge weights and soft priority scores as a pre-processing step, since the temperature β below (and its λ-side analogue α in Section 6) acts on their numerical scales.

For simplicity, we start with a simplified case where $\lambda_i \equiv 1$. For any π, its *total violation* against ω is $V_\omega(\pi) := \sum_{i\neq j} \omega_{ij} \cdot \mathbb{1}_{[j \text{ appears before } i \text{ in } \pi]}$. A Gibbs-style distribution that penalizes total violation is

$$p_\beta^\omega(\pi) \propto \exp\{-\beta \cdot V_\omega(\pi)\}, \tag{4}$$

where $\beta \geq 0$ is a *temperature* controlling the overall penalty strength. When ω is a DAG and $\beta \to \infty$, we have $p_\beta^\omega(\pi) = \text{Uniform}(\Pi^G)$, recovering PSV, a special case of PASV when $\lambda_i \equiv$ constant.

Next, we incorporate λ stage-wise, following PASV. Define the stage-wise violation cost $V_\omega(k; S_t) := \sum_{j\in S_t} \omega_{kj}$, i.e., the total strength of priorities that k would violate by being placed at position t. The **Generalized Priority-Aware Shapley Value (GPASV)** is defined as:

$$p^{(\lambda,\omega)}(\pi) \propto \prod_{t=1}^{n} \left[\underbrace{\frac{\lambda_{\pi_t} \exp\{-\beta \cdot V_\omega(\pi_t; S_t)\}}{\sum_{k\in S_t} \lambda_k \exp\{-\beta \cdot V_\omega(k; S_t)\}}}_{\text{(Part 1)}} \cdot \underbrace{\sum_{k\in S_t} \exp\{-\beta \cdot V_\omega(k; S_t)\}}_{\text{(Part 2)}} \right]. \tag{5}$$

At each stage t, Part 1 scores which player in S_t occupies position t: each $k \in S_t$ competes through its soft priority λ_k and its compliance with the pairwise priorities, measured by $V_\omega(k; S_t)$. Part 1 generalizes PASV's stage-wise softmax: in PASV, every player in $\max(S_t)$ is equally eligible to be π_t and competes solely through λ; in GPASV, the weighted graph ω no longer issues each player k just a pass/fail ticket, but instead a continuous eligibility score $\exp\{-\beta \cdot V_\omega(k; S_t)\}$ that discounts k by the total amount of pairwise priority violation it would cause if placed at position t; we naturally let this score rescale λ_k. Part 2 is the soft generalization of $|\max(S_t)|$ in (3). In PASV, $|\max(S_t)|$ is a key factor that ensures the reduction to PSV; under a weighted or cyclic ω, $\max(S_t)$ is no longer well-defined, so Part 2 instead sums the eligibility scores over S_t, which tends to $|\max(S_t)|$ when ω is a DAG and $\beta \to \infty$.

When $\lambda_i \equiv c$, it is not difficult to verify that (5) reduces to (4). When ω encodes a DAG with equal nonzero ω's and $\beta \to \infty$, (5) reduces to PASV (3); see Section 3.6 for formal limiting analysis.

3.3 Connections to Existing Literature

Gibbs distribution [5]. GPASV adopts a Gibbs-style encoding of pairwise priority weights; but due to the incorporation of λ, GPASV does not exactly fall into the Gibbs family.

Mallows model [33]. The Mallows model specifies a reference permutation $\pi_0 \in \Pi$ and a distribution $p_M(\pi) \propto \exp\{-d(\pi, \pi_0)\}$, where $d(\cdot, \cdot)$ is a dissimilarity measure, e.g., Kendall distance. Mallows is a *narrow special case* of GPASV: it does not consider λ and implicitly assumes that the priority graph can be induced by a single reference π_0 (in particular, ω must be acyclic).

Plackett-Luce model [36, 31]. When $\omega \equiv 0$, (5) reduces to a Plackett-Luce distribution with "worths" (λ_i), sampled stage-wise backward from position $t = n$ to $t = 1$.

3.4 Axiomatic Characterization of GPASV

Here, we outline the axiomatization, using some acronyms that will be defined later and in Appendix B. First, using Weber's axioms [42], we have E + L + NP + M ⇒ ROV – notice that GPASV is supported on the entire Π and does not use PASV's Maximal-Support (MS) axiom. Second, GSCF fixes a canonical stage-wise form for ROV distributions, rather than by itself imposing a new payoff-fairness requirement. Third, we engage boundary axioms: GWP and PVF to guarantee reduction to important special cases. Here, PASV's SCF, WP and EWU axioms are generalized to GSCF, GWP and PVF to suit generalized hard priority. Finally, ROV + GSCF + GWP + PVF ⇒ GPASV (see Theorem 1).

Now we carry out the outline. The first step (E+L+NP+M⇒ROV) is due to [42]. We start with GSCF.

Generalized state-choice factorization (GSCF). We use GSCF to specify this canonical stage-wise form, compatible with the sampling scheme in Section 3.2.

Definition 2 (GSCF)**.** *A distribution family $p_{\lambda,\omega}$ on Π satisfies GSCF if there exist a* state factor $s_\omega : 2^{[n]} \to \mathbb{R}_{\geq 0}$ *and a* choice factor $c_{\lambda,\omega}(\cdot\,;\cdot) : [n] \times 2^{[n]} \to [0,1]$*, s.t.*

$$p_{\lambda,\omega}(\pi) \propto \prod_{t=1}^{n} s_\omega(S_t) \cdot c_{\lambda,\omega}(\pi_t; S_t), \quad \pi \in \Pi,\ \forall(\lambda,\omega), \tag{6}$$

with $\sum_{i\in S} c_{\lambda,\omega}(i;S) = 1$ for every nonempty $S \subseteq [n]$.

GSCF is a canonical-form axiom: $s_\omega(S_t)$ is a prefix-dependent rescaling, and $c_{\lambda,\omega}(i;S_t)$ is the stage-wise choice weight of i for position t. Plackett-Luce is a familiar simple instance ($s \equiv 1$, $c = \lambda$-softmax over S_t); GPASV is another, where ω enters both factors. Specifically, Part 1 of (5) is the choice factor and Part 2 is the state factor; the two boundary axioms below pin down each of them. GSCF generalizes PASV's SCF [26, Definition 3.9] along two directions that reflect GPASV's weighted-graph nature: the choice factor ranges over the full S_t rather than the admissible subset $\max(S_t)$ (hard priority is no longer absolute), and the state factor depends on ω rather than on the DAG alone (recall Section 3.2).

Boundary axioms. Within the canonical GSCF family, we identify two important boundary cases.

Axiom 1 (Generalized Weight Proportionality (GWP))**.** *For every nonempty $S \subseteq [n]$,*

$$\frac{c_{\lambda,\omega}(i;S)}{c_{\lambda,\omega}(j;S)} = \frac{\lambda_i \exp\{-\beta \cdot V_\omega(i;S)\}}{\lambda_j \exp\{-\beta \cdot V_\omega(j;S)\}}, \quad \forall\, i,j \in S. \tag{7}$$

Equivalently, the log-odds of choosing i over j at stage t depends on λ_i/λ_j and $V_\omega(i;S) - V_\omega(j;S)$; PASV's Weight Proportionality (WP) axiom only encodes the former – the two coincide in the same limiting case under which GPASV reduces to PASV (Section 3.2). In PASV, WP is the key to containing WSV as a boundary case; together with $\sum_{i\in S} c_{\lambda,\omega}(i;S) = 1$, GWP fixes the choice factor to Part 1 of (5).

Axiom 2 (Pairwise-Violation Factorization (PVF))**.** *If $\lambda_i \equiv 1$, then $p(\pi) \propto \exp\{-\beta \cdot V_\omega(\pi)\}$.*

That is, when players' soft priority weights λ_i's are equal, p recovers the Gibbs-style form (4). We use the Gibbs form as the "no soft priority" boundary for two reasons: it naturally extends PASV's EWU to weighted and cyclic graphs, and given the expected total violation, it is the maximum-entropy (least informative) distribution on Π. Together with GWP, PVF fixes the state factor to Part 2.

Theorem 1 (Uniqueness of GPASV)**.** *The only ROV satisfying GSCF + GWP + PVF is GPASV.*

3.5 Diagnostic Tool: Priority Sweeping

Soft priority sweep (inherited from [26]). As pointed out by [26], soft priority λ is often unavailable in practice, since relevant metadata such as trustworthiness, compliance risk, or originality scores are not routinely collected. Following [26], we equip GPASV with *priority sweeping*: one can vary a single λ_i over $(0,\infty)$ to diagnose whether the unknown soft priority materially affects the valuation. This inherits the *idea* from [26] but not the algorithm: since GPASV's stage-wise softmax acts on the full S_t rather than the admissible subset $\max(S_t)$, the computation needs adaptation (see Section 4).

Hard priority sweep (new to GPASV). GPASV's weighted graph ω opens a sweeping axis that PASV's binary DAG does not admit. The two extremes are $\beta = 0$ (no pairwise priority) and $\beta \to \infty$ (hard priority; see Section 3.6), and sweeping intermediate β traces the transition between them.

Overall, priority sweeping should report λ-only, ω-only, and joint sweeping: this reveals robustness and flags possible double-counting when λ aligns with priority graph; see Sections 5–6.

Choosing the sweep range. The sweeping parameters (e.g., β, and α in Section 6) are not tuning parameters: in Shapley-type valuation, the valuation itself is usually the final output, so there is no downstream task that can serve as an undisputed judge for choosing them [41]. Instead, they encode the evaluator's preference on how to weigh the two priority sources, and GPASV's deliverable is the family of valuations along the sweep with its sensitivity diagnostics, rather than a single number. In practice, we sweep each temperature from 0 upward until the valuation stabilizes, and treat that value as a proxy for ∞.

3.6 Limiting-Case Analysis

Let $G_\omega = ([n], \mathcal{E}_\omega)$ with $\mathcal{E}_\omega = \{(i,j) : \omega_{ij} > 0\}$ denote the directed graph induced by ω.

Theorem 2 (Hard-penalty limit)**.** *For every fixed λ and ω, we have*

$$\lim_{\beta\to\infty} p^{(\lambda,\omega)}(\pi) \propto \mathbb{1}_{[\pi\in\widetilde{\Pi}^{G_\omega}]} \cdot \prod_{t=1}^{n} \frac{\lambda_{\pi_t}\,|M_\omega(S_t)|}{\sum_{k\in M_\omega(S_t)} \lambda_k}, \tag{8}$$

where recall $V_\omega(\pi)$ and $V_\omega(k;S)$ from Section 3.2, and define $\widetilde{\Pi}^{G_\omega} := \arg\min_{\pi\in\Pi} V_\omega(\pi)$ and $M_\omega(S) := \arg\min_{k\in S} V_\omega(k;S)$.

In Theorem 2, $\widetilde{\Pi}^{G_\omega}$ collects permutations with minimum total violation; while $M_\omega(S)$ is the soft analogue of $\max(S)$, it collects the members of S whose placement at the end of S incurs the least total violation within S. If G_ω is a DAG: $\widetilde{\Pi}^{G_\omega} = \Pi^{G_\omega}$, $M_\omega(S_t) = \max(S_t)$ for prefix S_t, and Theorem 2 shows that GPASV reduces to PASV (3) on G_ω. But if G_ω is cyclic, the limit in Theorem 2 generally does not reduce to any PASV. A simple example is a big cycle (equal edge weights). Appendix D gives a sharper example: support becomes DAG, but distribution is still not PASV.

4 Computation

Recall that GPASV is defined by (2) and (5). Naturally, we discuss three main topics: (i) how to sample π from (5); (ii) how to compute (2); (iii) acceleration tricks. Due to page limit, here we only present the gist and relegate detailed algorithms, and auxiliary derivations to Appendix F.

Sampling π. We draw π from $p^{(\lambda,\omega)}$ (5) using an adjacent-swap Metropolis–Hastings (MH) chain [22, 4, 26]. At each iteration, propose to swap (π_i, π_{i+1}) with acceptance probability $\min\{1, p^{(\lambda,\omega)}(\pi^{\text{swap}})/p^{(\lambda,\omega)}(\pi)\}$. It turns out that for GPASV, almost all factors in this ratio cancel and only a local expression remains, making computation efficient (see Proposition 1). Initialization for this MCMC requires extra care. PASV can conveniently start from any valid linear extension, but for GPASV, when ω is cyclic, no linear extension exists; a cold start can waste many burn-in iterations. To address this challenge, we devise a dedicated stage-wise greedy method (see Algorithm 2): sample players backward from position n, each with probability proportional to a term that turns out to be exactly the GSCF choice factor $c_{\lambda,\omega}$ in (6) instantiated for GPASV, see Appendix F.1.2. Mixing time bounds for this chain are generally intractable; we study mixing empirically in Section 5.

Estimating the expectation in (2). Given sampled permutations, the most straightforward estimator simply replaces the expectation in (2) by a Monte Carlo average, which we use as our default. In Appendix F.2.2, we additionally describe a surrogate-assisted variant adapted from [30, 32, 15, 44]: first fit a cheap surrogate $\widehat{h} \approx U$, whose GPASV is usually easier to compute; second, bias-correct for using $\widehat{h}$ in lieu of U. This method works when $\widehat{h}$ well-captures U; otherwise, the extra surrogate-fitting cost can outweigh its merit.

Further acceleration. A simple trick allows substantial cost reduction with minimal implementation effort: we cache $U(S)$ the first time it is evaluated and reuse the cached value thereafter. A separate acceleration is specific to priority sweeping (Section 3.5), where the target $p^{(\lambda,\omega)}$ moves along a trajectory. Since nearby targets along this trajectory are similar, we may largely reuse the previous permutation sample instead of redrawing, via *self-normalized importance sampling* (SNIS) [18], and monitor to ensure sufficient *effective sample size*. See Appendix F.3 for details.

5 Empirical Validation

We conduct three simulations to assess the accuracy of our GPASV, test speed-up tricks, and compare its priority sweeping results with its counterpart from the previous work PASV to deepen our understanding.

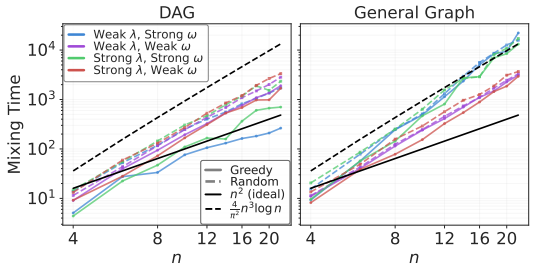


Figure 2: Simulation 1: mixing time, greedy (solid) vs random (dashed) initializations.

Simulation 1: mixing time of MCMC. Recall from Section 4 that GPASV requires sampling from a non-uniform distribution on permutations; the first step is to burn-in the MH chain until stationarity; the number of iterations needed is called *mixing time*. We diagnose mixing by the accuracy of pairwise-order probabilities $\mathbb{P}(\{i \text{ appears before } j\})$ and tested both DAG and general graphs. Due to page limit, here we only plot for a representative setting and relegate all details to Appendix G.1. Figure 2 suggests that the mixing speed of our algorithm is competitive compared to literature. In particular, we find that our greedy initialization method (Algorithm 2) can significantly speed up mixing.

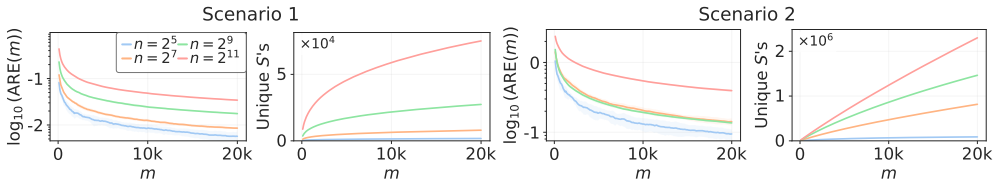


Figure 3: Accuracy and acceleration by caching (case 2); full grids are in Appendix G.2.

Simulation 2: Monte Carlo accuracy and surrogate-assisted acceleration. After the MH chain mixes well, the next question is the accuracy of the Monte Carlo estimation of (2). To separate sampling error from modeling error, we use two synthetic game families with closed-form GPASV values. Scenario 1 uses a line DAG: $1 \to \cdots \to n$; Scenario 2 has the structure of a *DAG of cycles*: players are partitioned into blocks, the blocks form a DAG among themselves, and each block is internally a big directed cycle.

Figure 3 shows that the error of direct MC quickly decreases with sample budget; the problem's difficulty increases with n (# of players), decreases for graphs with stronger priority structures; caching $U(S)$ significantly reduces computational cost. In Figure 4 and Table 1, we compared (i) our direct MC estimator on (2) to two surrogate-assisted estimators introduced in F.2.2 using (ii) linear; and (iii) quadratic surrogates – roughly speaking, (ii) fits $U(S) \approx U(\emptyset) + \sum_{i\in S} a_i$, while (iii) fits $U(S) \approx U(\emptyset) + \sum_{i\in S} a_i + \sum_{i,j\in S} b_{i,j}$.

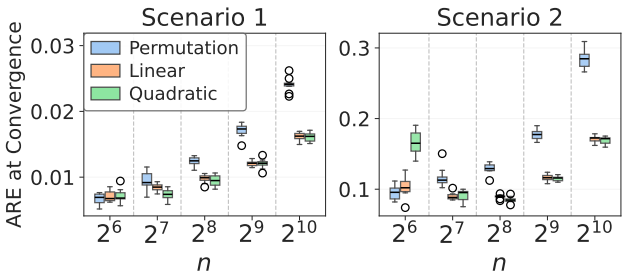


Figure 4: Surrogate-assisted methods (case 2). Full results are in Appendix G.2.

Table 1: Runtime (seconds) and training memory (GB) under matched utility budgets (case 2). Values are mean (std.dev.). Permutation reports runtime only. Full results are in Appendix G.2.

	Scenario 1					Scenario 2				
	Permutation	Linear		Quadratic		Permutation	Linear		Quadratic	
n	Time (s)	Time (s)	Mem (GB)	Time (s)	Mem (GB)	Time (s)	Time (s)	Mem (GB)	Time (s)	Mem (GB)
64	4.94 (0.01)	26.51 (0.15)	0.001 (0.000)	26.49 (0.10)	0.002 (0.000)	4.69 (0.02)	35.50 (0.38)	0.093 (0.000)	36.08 (0.24)	0.290 (0.001)
128	8.53 (0.02)	46.72 (0.19)	0.002 (0.000)	46.89 (0.26)	0.012 (0.000)	8.15 (0.04)	61.27 (0.28)	0.194 (0.000)	63.95 (0.22)	0.986 (0.003)
256	15.94 (0.03)	91.73 (0.17)	0.009 (0.000)	92.30 (0.28)	0.081 (0.001)	15.99 (0.07)	120.17 (0.39)	0.592 (0.000)	138.65 (0.88)	5.620 (0.008)
512	36.45 (0.16)	191.07 (0.65)	0.031 (0.000)	194.82 (0.74)	0.566 (0.006)	34.36 (0.06)	231.47 (1.18)	1.158 (0.001)	317.75 (3.33)	21.000 (0.027)
1024	195.76 (2.46)	663.81 (6.91)	0.106 (0.001)	700.01 (6.31)	3.724 (0.031)	205.55 (0.51)	725.34 (4.64)	2.312 (0.002)	1378.02 (9.58)	108.276 (0.155)

Key takeaway: while surrogate methods improve estimation accuracy in most cases, they (especially the quadratic method) could introduce remarkable runtime and memory overhead. These additional costs can be much more significant than PASV, since PASV can zero out $b_{i,j}$'s corresponding to DAG edges, while under GPASV this is impossible. Practically, we recommend direct MC in most cases, and the quadratic surrogate only when (a) n is small enough to afford its $O(n^2)$ terms $b_{i,j}$, or (b) ω is (nearly) a DAG and a large β is preferred, i.e., close to PASV's setting.

Simulation 3: priority sweeping. This simulation compares the impact of the soft priority λ_i under GPASV and PASV. Recall from [26] that under PASV, the impact of λ_i is limited by the hard priority. However, since GPASV has softened the hard priority, any permutation is possible. To stabilize results, here we pick a subset of size $n/2$ and set their λ_i's to a common value λ_0 and sweep. Figure 5 reports the mean rank and valuation of this group. The valuations use two utilities indexed by random player subsets T (Appendix G.3): the sum-of-unanimity (SOU) utility credits T if *all* members of T are in S, and the sum-of-race (SOR) utility if *any* member is; SOU thus rewards late-positioned players and SOR early-positioned ones. When β is small, λ_0 moves the results more under GPASV than under PASV (black curve); this is a direct consequence of GPASV's full support on Π rather than a separate finding, and the gap closes as $\beta \to \infty$. When $\beta = 0$, GPASV recovers Plackett-Luce, in which scenario λ's impact is maximized.

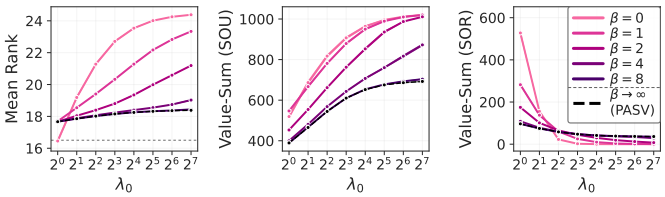


Figure 5: Simulation 3: priority sweeping.

6 Application: LLM Evaluation under Human Preference Graphs

We apply GPASV to LLM ensemble valuation, using Chatbot Arena's [7] pairwise human preferences as a cyclic, weighted priority graph that PASV cannot handle. Aggregated human preferences over LLMs are known to contain cycles with high probability [29], so a cycle-aware method is not a corner case but an essential ingredient for this application. The main takeaway is that valuation should not be treated as a single, best-chosen number automatically read off from the data. Instead, the user must decide how to weigh soft, individual priorities against pairwise user-preference priorities, and this choice materially affects the eventual valuation.

Data and setup. Our goal is to apply GPASV to value each candidate LLM. Running GPASV on this task requires three inputs, each sourced from a public benchmark: a coalition utility, a pairwise priority graph, and a soft priority. We restrict attention to the 20 models that appear in both MT-Bench [46] and Chatbot Arena [7], so each player is one model and a coalition S is a subset of these models.

Coalition utility. The coalition utility is supplied by MT-Bench [46], which consists of 80 two-turn prompts together with released responses from a large set of models scored by an LLM-as-a-judge protocol on a 1–10 scale. For a coalition S and a prompt q, we first pass the first-turn responses of models in S through a third-party LLM aggregator to synthesize a single ensemble answer, then ask a third-party LLM judge to score that answer against the MT-Bench protocol; we repeat the aggregator-judge pipeline for the second turn, conditioned on the first-turn dialogue so that coherence is scored as well. The prompt-level utility is the mean of the two turn scores, and the coalition utility $U_{\text{ens}}(S)$ is the mean over all 80 prompts, with $U_{\text{ens}}(\emptyset) = 0$. Both aggregator and judge are

instantiated by `Qwen3.5-35B-A3B-fp8` [37]. To check the judge choice, we compare our judge scores with the released GPT-4 judge scores of MT-Bench on the 20 singleton coalitions, where no aggregator is involved: the Pearson correlations are 0.984, 0.983, and 0.986 for Turn 1, Turn 2, and their mean, respectively; see Appendix A for the scope of this check. Further details are relegated to Appendix H.2; templates are in Appendix H.6.

Pairwise priority graph. Chatbot Arena [7] supplies pairwise human preferences. For each model pair (i, j) with at least 50 recorded comparisons, let i denote the majority-preferred model and set $\omega_{ij} = \widehat{p}_{ij} - 1/2$ and $\omega_{ji} = 0$, where $\widehat{p}_{ij}$ is the empirical win probability of i over j. The graph has cycles and is not a DAG [29]. The temperature β in (5) controls how strongly this graph shapes the GPASV order distribution: $\beta = 0$ turns off the priority graph entirely, leaving only λ (SV if all λ_i's are equal); larger β enforces the Arena majority directions more strongly.

Soft priority. The soft priority encodes a deployment preference for open-source models over paid proprietary APIs. Let $z_i = 1$ if model i is open-source and $z_i = 0$ otherwise (the paid models in our set are `gpt-4`, `gpt-3.5-turbo`, `claude-v1`, `claude-instant-v1`, and `palm-2`). A non-negative temperature α converts this raw preference into the player weights $\lambda_i = \exp(-\alpha z_i)$ that enter GPASV in (5), so that $\alpha = 0$ recovers $\lambda_i \equiv 1$ (no soft priority). Because GPASV samples backward, larger α places open-source models earlier in the sampled permutation. This is not a claim that open-source models are intrinsically better; it simply reflects an evaluator who wants to know how much of the ensemble's performance open-source models are responsible for before reaching for proprietary APIs.

Experimental design. Our experiments scan (α, β) along three one-dimensional slices: α varies alone (with $\beta = 0$), β varies alone (with $\alpha = 0$), and the two vary together with $\alpha = \beta$. Along each slice the non-zero temperature takes the values $\{1, 2, 4, 8, 16, 32\}$, and all three slices share the baseline $(\alpha, \beta) = (0, 0)$, at which GPASV reduces to the classical Shapley value with no priority applied. This gives 19 settings in total. The α-only and β-only slices are ablations for diagnosing whether the soft priority and the priority graph encode overlapping signals, while the joint slice studies the evaluator's deliberate decision to combine them. Due to space, the main paper presents a few representative (α, β) pairs; full results across all 19 settings are in Appendix H.4.

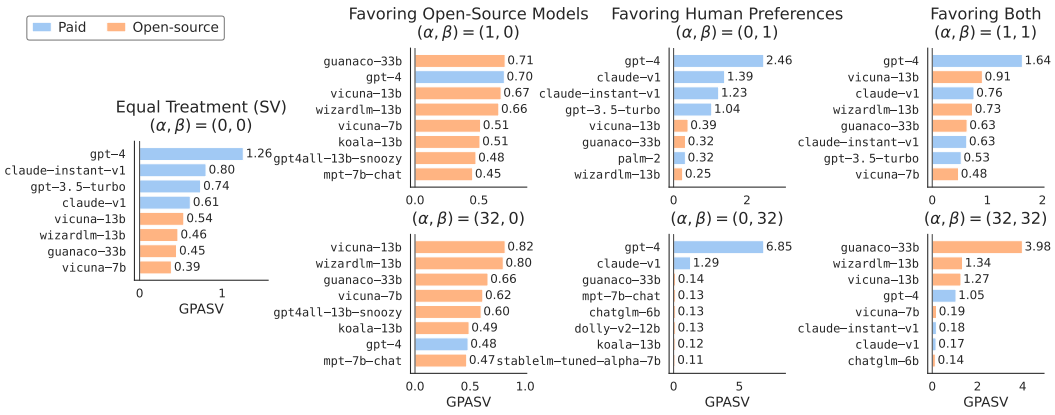


Figure 6: Top-valued models under different priority balances. Colors: paid vs open-source.

Computation. The LLM inference cost in utility evaluations dominates that of permutation sampling: one un-cached $U_{\text{ens}}(S)$ evaluation costs $4 \times 80 = 320$ calls. Naively evaluating every prefix over 19 settings would require $O(10^8)$ LLM calls. We use the two accelerations from Section 4: cache each (S, q) aggregator-judge output, and reuse neighboring-setting permutations by SNIS, drawing fresh samples only when effective sample size is low. Appendix H.3 reports a realized 5.4× reduction in distinct utility evaluations.

Results. Figures 6 and 7 report the scan outcomes: top-8 GPASV values under 7 representative (α, β) settings, and group-sum GPASV along the three slices. Four observations.

1. Paid dominates the no-priority baseline. At $(0, 0)$, GPASV reduces to the Shapley value, and paid models hold ranks 1–4 led by `gpt-4` at 1.26; the first open-source model only appears at rank 5. The paid group-sum exceeds the open-source group-sum, even though paid contains only 5 of the 20 models; equivalently, the paid per-model average is substantially higher. This skew comes from the coalition utility alone — the aggregator-and-judge pipeline simply scores paid responses higher. All later observations are to be read against this baseline.

2. α lifts the favored group uniformly. Along the $\beta = 0$ slice, the paid and open-source group-sum curves cross near $\alpha \approx 2$ and then separate sharply, with open-source climbing and paid collapsing (Figure 7, left). Even at $\alpha = 1$, `gpt-4` has lost rank 1 to `guanaco-33b` and the other paid models fall out of the top 8; at $\alpha = 32$ every open-source model is lifted by a similar multiplicative factor, because α enters each $\lambda_i = \exp(-\alpha z_i)$ independently.

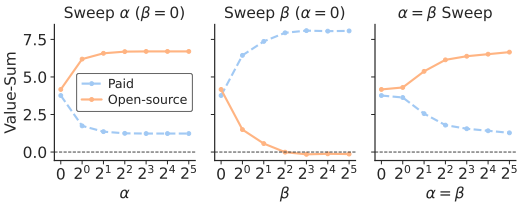


Figure 7: Group-sum GPASV for paid and open-source models across priority sweeps. The α sweep strengthens open-source soft priority, the β sweep strengthens the preference graph priority, and the joint sweep increases both together.

3. β concentrates GPASV on a single preference-graph hub. Along the $\alpha = 0$ slice, increasing β does not spread the gain across paid models. At $\beta = 32$, `gpt-4` alone attains GPASV value 6.85 over $5\times$ the runner-up, while most other paid models fall to the open-source tail (~ 0.1). The paid group-sum in Figure 7 (middle) rises, but is carried by this one model rather than spread across the five. The structural reason is that β acts through the pairwise violation V_ω, which couples every player to every other, so graph-theoretic hubs absorb attribution rather than uniform category members. For `gpt-4` specifically, two factors reinforce this: it wins most of its Arena comparisons, which places it early in almost all sampled permutations, and it is highly effective on its own, leaving little room for later players to contribute.

4. On the joint slice, α wins. The joint slice $\alpha = \beta$ could a priori track either axis or interpolate between them; empirically it tracks the α-only axis (Figure 7, right). At $(32, 32)$, `guanaco-33b` leads at 3.98, while `gpt-4` drops to rank 4 at 1.05 — nowhere near the 6.85 it commands under β-only. The player-level soft priority, which operates through an additive log-factor for every player, dominates the edge-level graph priority, which concentrates into a small number of hubs.

Finally, under a bootstrap of the Arena comparisons, the GPASV rankings remain stable for $\beta \leq 16$ (mean Kendall $\tau \geq 0.91$), whereas ranking-then-PASV baselines that first project the graph to a Bradley–Terry or round-robin total order disagree substantially with GPASV and with each other; see Appendix H.5.

7 Key Takeaway and Limitations

Take-home message. Priority-aware valuation is **not a one-button operation**: whenever both player-level and pairwise priority information are present (e.g., open-source deployment preference + Chatbot Arena preference graph), the user must decide how to balance the two sources, and the resulting valuation can materially depend on that decision. This is **not a method-level inconsistency but a natural property of the problem**: unless an external objective is specified, the data alone cannot identify a uniquely correct priority trade-off; different evaluator preferences define different coherent valuations. The value of GPASV is to keep these modeling choices visible: it exposes the priority trade-off as an explicit, sweepable input and accepts cyclic priority graphs without first reducing them to a DAG.

Limitations. We discuss two main limitations: utility calls and priority graph stability. Due to page limit, we relegate the detailed discussion to Appendix A.

Acknowledgments and Disclosure of Funding

Lee and Zhang were supported by NSF DMS-2311109 and DMS-2610424. Liu and Tang were supported by NSF DMS-2412853, DMS-2610423, and Jane Street Group, LLC.

References

[1] J. F. Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19: 317, 1964.

[2] D. Black et al. The theory of committees and elections. 1958.

[3] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[4] R. Bubley and M. Dyer. Faster random generation of linear extensions. *Discrete mathematics*, 201(1-3):81–88, 1999.

[5] G. T. Cantwell and C. Moore. Belief propagation for permutations, rankings, and partial orders. *Physical Review E*, 105(5):L052303, 2022.

[6] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & operations research*, 36(5):1726–1730, 2009.

[7] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M. Jordan, J. E. Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

[8] J. A. M. N. C. Condorcet et al. Essai sur l'application de l'analyse à la probabilité des décisions, rendues à la pluralité des voix/l c par m. le marquis de condorcet..., 1785.

[9] J. Derks. A new proof for weber's characterization of the random order values. *Mathematical Social Sciences*, 49(3):327–334, 2005.

[10] P. Dubey, A. Neyman, and R. J. Weber. Value theory without efficiency. *Mathematics of Operations Research*, 6(1):122–128, 1981.

[11] U. Faigle and W. Kern. The shapley value for cooperative games under precedence constraints. *International Journal of Game Theory*, 21(3):249–266, 1992.

[12] P. C. Fishburn. Condorcet social choice functions. *SIAM Journal on applied Mathematics*, 33 (3):469–489, 1977.

[13] C. Frye, C. Rowat, and I. Feige. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. *Advances in neural information processing systems*, 33: 1229–1239, 2020.

[14] F. Fumagalli, L. Butler, J. S. Kang, K. Ramchandran, and R. T. Witter. An odd estimator for shapley values. *arXiv preprint arXiv:2602.01399*, 2026.

[15] F. Fumagalli, R. T. Witter, and C. Musco. PolySHAP: Extending kernelSHAP with interaction-informed polynomial regression. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=M19J8UGguq.

[16] A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

[17] M. E. Glickman. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 48(3):377–394, 1999.

[18] T. Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995.

[19] A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11): 558–562, 1962.

[20] E. Kalai and D. Samet. On weighted Shapley values. *International journal of game theory*, 16 (3):205–222, 1987.

[21] K. Kangas, T. Hankala, T. M. Niinimäki, and M. Koivisto. Counting linear extensions of sparse posets. In *IJCAI*, volume 16, pages 603–609, 2016.

[22] A. Karzanov and L. Khachiyan. On the conductance of order markov chains. *Order*, 8(1):7–15, 1991.

[23] A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

[24] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[25] Y. Kwon and J. Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049*, 2021.

[26] K. Lee, Z. Liu, W. Tang, and Y. Zhang. Priority-aware shapley value. *arXiv preprint arXiv:2602.09326*, 2026.

[27] W. Li and Y. Yu. Robust data valuation with weighted banzhaf values. *Advances in Neural Information Processing Systems*, 36:60349–60383, 2023.

[28] W. Li and Y. Yu. One sample fits all: Approximating all probabilistic values simultaneously and efficiently. *Advances in Neural Information Processing Systems*, 37:58309–58340, 2024.

[29] K. Liu, Q. Long, Z. Shi, W. J. Su, and J. Xiao. Statistical impossibility and possibility of aligning llms with human preferences: From condorcet paradox to nash equilibrium. *arXiv preprint arXiv:2503.10990*, 2025.

[30] Z. Liu, K. Lee, Y. Zhang, and W. Tang. First-order efficiency for probabilistic value estimation via a statistical viewpoint. *arXiv preprint arXiv:2605.02827*, 2026.

[31] R. D. Luce et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.

[32] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[33] C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[34] NeurIPS. Neurips 2026 main track handbook, 2026. URL https://neurips.cc/Conferences/2026/MainTrackHandbook.

[35] A. S. Nowak and T. Radzik. On axiomatizations of the weighted shapley values. *Games and Economic Behavior*, 8(2):389–405, 1995.

[36] R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[37] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[38] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2, 1953.

[39] T. Talvitie, T. M. Niinimäki, and M. Koivisto. The mixing of markov chains on linear extensions in practice. In *IJCAI*, pages 524–530, 2017.

[40] J. T. Wang and R. Jia. Data Banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6388–6421. PMLR, 2023.

[41] J. T. Wang, T. Yang, J. Zou, Y. Kwon, and R. Jia. Rethinking data shapley for data selection tasks: Misleads and merits. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[42] R. J. Weber. *Probabilistic values for games*, page 101–120. Cambridge University Press, 1988.

[43] D. B. Wilson. Mixing times of lozenge tiling and card shuffling markov chains. *The Annals of Applied Probability*, 14(1):274–325, 2004.

[44] R. T. Witter, Y. Liu, and C. Musco. Regression-adjusted monte carlo estimators for shapley values and probabilistic values. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Qabko39AS5.

[45] Y. Xu, L. Ruis, T. Rocktäschel, and R. Kirk. Investigating non-transitivity in llm-as-a-judge. *arXiv preprint arXiv:2502.14074*, 2025.

[46] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[47] X. Zheng, Y. Huang, X. Chang, R. Jia, and Y. Tan. Rethinking data value: Asymmetric data Shapley for structure-aware valuation in data markets and machine learning pipelines. *arXiv preprint arXiv:2511.12863*, 2025.

Computer Code

Computer code is available at: https://github.com/KiljaeL/GPASV

A Limitations

A.1 Utility Calls

The main computational cost of GPASV lies in repeated evaluations of the coalition utility $U(S)$. This cost is shared by essentially all model-agnostic data valuation methods in the random-order family, and is, in our view, unavoidable: any method that treats U as a black box has no recourse but to query it, and accurate estimation of marginal contributions imposes a minimum query budget that scales with the number of players. The cost becomes especially salient when each query to U is itself expensive, as in the LLM ensemble setting of Section 6, where evaluating $U(S)$ for a single coalition requires running an LLM-as-an-aggregator and an LLM-as-a-judge over all 80 MT-Bench questions, with 20 candidate models giving 2^{20} possible coalitions.

To address this limitation, our framework incorporates three computational devices that work jointly with the GPASV definition rather than being bolted on after the fact. Subset caching (Section 4) exploits the fact that the same subset S recurs many times across both the permutation-based estimator and the surrogate-based estimator, so each distinct $U(S)$ is evaluated at most once. Self-normalized importance sampling (Section F.3.2) reuses permutations already drawn under one (λ, ω) to estimate GPASV at neighboring parameter values, which is what makes the priority sweeps in Section 6 affordable. The two-stage surrogate-adjusted residual estimator (Section F.2.2) further reduces variance per utility query by absorbing most of the signal into a fitted surrogate and spending the remaining query budget on residual correction. Together, these devices allow the full LLM application of Section 6, including its priority sweeps, to be carried out within a fixed and modest LLM-call budget.

The same cost also limits how far we can vary the utility pipeline itself. The judge check in Section 6 uses singleton utilities only, which isolates the choice of judge but not that of aggregator, and its conclusion is specific to our aggregator–judge pipeline. Full component-replacement ablations, i.e., recomputing coalition utilities and GPASV under alternative judges and aggregators, require recomputing all utilities, and we leave them as an important direction for future work.

A.2 Priority Graph

A second limitation concerns the priority graph itself. Throughout this paper we treat the priority graph as a given input and study how it shapes the resulting valuation, but in practice the graph is often estimated from finite, noisy, or partially observed data, as is the case for the Chatbot Arena pairwise preferences used in Section 6. At finite β, GPASV remains a smooth function of the edge weights, but the $\beta \to \infty$ hard-penalty boundary can amplify near-ties in violation cost. Hence priority sweeps can diagnose empirical instability, but they are not formal statistical error bars. Deriving finite-sample error bounds or sharper stability diagnostics for noisy estimated graphs is beyond the present work and is an important direction for future research. As a first step, Appendix H.5 reports a graph bootstrap on the Chatbot Arena graph: the GPASV rankings are stable for moderate β, and become less stable at large β, where the effective sample size of importance reweighting is also low.

B Background for the Shapley Value and Random Order Values

This appendix collects the standard axioms invoked throughout the paper. We fix the player set $[n]$ and consider utility functions $U : 2^{[n]} \to \mathbb{R}$ with $U(\emptyset) = 0$. A *value* is a mapping ψ assigning to each such U a payoff vector $\psi(U) = (\psi_i(U))_{i\in[n]}$.

The following four axioms are originally due to Shapley [38].

Axiom 3 (Efficiency (E))**.** $\sum_{i\in[n]} \psi_i(U) = U([n])$.

Axiom 4 (Linearity (L))**.** *For utilities U, V and scalars $\alpha, \beta \in \mathbb{R}$, $\psi(\alpha U + \beta V) = \alpha\psi(U) + \beta\psi(V)$.*

Axiom 5 (Null Player (NP))**.** *If player i contributes nothing, i.e., $U(S \cup \{i\}) = U(S)$ for all $S \subseteq [n] \setminus \{i\}$, then $\psi_i(U) = 0$.*

Axiom 6 (Symmetry (S)). *If two players i, j are interchangeable, i.e., $U(S \cup \{i\}) = U(S \cup \{j\})$ for all $S \subseteq [n] \setminus \{i, j\}$, then $\psi_i(U) = \psi_j(U)$.*

E says the grand-coalition revenue $U([n])$ is fully distributed; L says the value commutes with affine combinations of utilities; NP rules out paying a player whose marginal contribution is identically zero; S enforces anonymity by treating exchangeable players identically. Shapley [38] shows that E + L + NP + S uniquely characterize the Shapley value ψ^{SV} in (1).

For random order values, anonymity is precisely what one wants to relax in order to encode priority. Weber [42] replaces S with the following monotonicity (M) axiom and obtains a strictly larger class.

Axiom 7 (Monotonicity (M)). *If U is monotone in the sense $S \subseteq T \Rightarrow U(S) \leq U(T)$, then $\psi_i(U) \geq 0$ for every $i \in [n]$.*

M rules out negative payoffs whenever larger coalitions never reduce revenue. Weber [42] shows that a value satisfies E + L + NP + M if and only if it admits the ROV form (2) for some distribution p over Π. In other words, ROV is exactly the family obtained by deliberately relaxing Symmetry: it preserves the additive, contribution-based fairness ideals (E, L, NP, M) while opening room to treat players asymmetrically through the choice of permutation distribution p.

C Two Equivalent Representations of GPASV

The definition of GPASV in (5) encodes pairwise priorities based on a Gibbs-style representation with *additive* priorities $\omega_{ij} \geq 0$. However, the same family of distributions can be expressed in an alternative, *multiplicative* representation. This is only a shorthand that connects more directly to the numerical implementation and the algorithms in Appendix F. Define

$$\widetilde{\omega}_{ij} = \exp(-\beta \omega_{ij}).$$

Note that the temperature β is absorbed in the multiplicative representation. By one-to-one correspondence, $\widetilde{\omega}_{ij} \in (0, 1]$ and $\widetilde{\omega}_{ij} = 1$ corresponds to the absence of pairwise priority. In this representation, the stepwise violation factor is the product

$$\widetilde{\omega}_k^S := \prod_{j \in S \setminus \{k\}} \widetilde{\omega}_{kj}, \quad S \subseteq [n],\ k \in [n], \tag{9}$$

with $\widetilde{\omega}_{kk} = 1$, so that we have the equivalence

$$\exp\{-\beta \cdot V_\omega(k; S)\} = \prod_{j \in S} \exp(-\beta \omega_{kj}) = \prod_{j \in S} \widetilde{\omega}_{kj} = \widetilde{\omega}_k^S. \tag{10}$$

Under this equivalence, the GPASV distribution (5) admits the equivalent product form

$$p^{(\lambda,\omega)}(\pi) \propto \prod_{t=1}^{n} \left[\frac{\lambda_{\pi_t} \widetilde{\omega}_{\pi_t}^{S_t}}{\sum_{k \in S_t} \lambda_k \widetilde{\omega}_k^{S_t}} \cdot \sum_{k \in S_t} \widetilde{\omega}_k^{S_t} \right]. \tag{11}$$

Reading off (11), GPASV's GSCF factorization (Definition 2) instantiates as

$$c_{\lambda,\omega}(i; S) = \frac{\lambda_i \widetilde{\omega}_i^S}{\sum_{k \in S} \lambda_k \widetilde{\omega}_k^S}, \quad s_\omega(S) = \sum_{k \in S} \widetilde{\omega}_k^S. \tag{12}$$

That is, the choice factor is a λ-weighted softmax over S with weights $\widetilde{\omega}_k^S$, and the state factor is the corresponding partition sum.

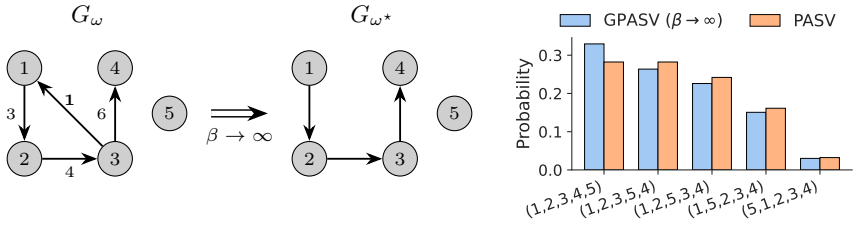
Table 2 summarizes the relationship between the two representations.

D A Detailed Example of the GPASV under Cyclic Graph

Here we provide a detailed analysis of the cyclic counterexample where GPASV does not reduce to PASV, as mentioned in Section 3.6. Consider five players with node weights $\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 3, \lambda_4 = 4, \lambda_5 = 5$ and directed edge weights $\omega_{12} = 3, \omega_{23} = 4, \omega_{31} = 1, \omega_{34} = 6$ (player 5 isolated), see Figure 8.

Table 2: Correspondence between the multiplicative representation $\widetilde{\omega}_{ij} = \exp(-\beta\omega_{ij})$ and the additive representation ω_{ij} used in the main text.

Quantity/Case	Correspondence
Pairwise priority	$\widetilde{\omega}_{ij} = \exp(-\beta\omega_{ij})$
Stepwise violation	$\widetilde{\omega}_k^S = \exp(-\beta \cdot V_\omega(k; S))$
No priority	$(\widetilde{\omega}_{ij} = 1) \sim (\omega_{ij} = 0)$
Hard priority	$(\widetilde{\omega}_{ij} \to 0) \sim (\omega_{ij}$ or $\beta \to \infty)$



(a) (b)

Figure 8: Example showing that cyclic GPASV does not reduce to PASV; (a) the support of the limiting GPASV distribution collapses to the feasible permutations of the DAG obtained by deleting the minimum-violation edge; (b) Yet the unnormalized distribution of the limiting GPASV differs from that of the PASV on the reduced DAG.

If we delete the edge $3 \to 1$, the graph becomes acyclic, and the permutations obeying the remaining edges achieve the minimum total violation of $\omega_{31} = 1$. Write $\omega^\star$ for the edge weights of the reduced DAG, so that $\omega^\star_{31} = 0$ and $\omega^\star_{ij} = \omega_{ij}$ for all other pairs. Then, the support of the limiting GPASV matches the support of the PASV on $G^\star_\omega$:

$$\widetilde{\Pi}^{G_\omega} \;=\; \Pi^{G^\star_\omega} \;=\; \big\{(1,2,3,4,5),\,(1,2,3,5,4),\,(1,2,5,3,4),\,(1,5,2,3,4),\,(5,1,2,3,4)\big\}.$$

It is enough to compare with $G^\star_\omega$: if the limiting GPASV law coincided with PASV on some DAG H, then its support would satisfy $\Pi^H = \widetilde{\Pi}^{G_\omega} = \Pi^{G^\star_\omega}$. The set of linear extensions determines the same partial order up to transitive closure, and PASV depends only on this induced partial order. Thus PASV on H coincides with PASV on $G^\star_\omega$, and it suffices to show that the limiting GPASV law differs from PASV on $G^\star_\omega$.

Since the two supports agree, we may compare the unnormalized distributions of the limit of GPASV in Theorem 2 with the PASV (3) on $G_{\omega^\star}$, which are given by:

$$\lim_{\beta\to\infty} \widetilde{p}^{(\lambda,\omega)}(\pi) \;=\; \prod_{t=1}^{n} \frac{\lambda_{\pi_t}\,|M_\omega(S_t)|}{\sum_{k\in M_\omega(S_t)}\lambda_k}, \quad \widetilde{p}^{(\lambda,G_{\omega^\star})}(\pi) \;=\; \prod_{t=1}^{n} \frac{\lambda_{\pi_t}\,|\max(S_t)|}{\sum_{k\in \max(S_t)}\lambda_k}.$$

For a permutation $\pi = (1,2,3,4,5)$, their ratio (PASV/GPASV) is 1. However, on the permutation $\pi = (1,2,3,5,4)$, their ratio is

$$\frac{\text{PASV}}{\text{GPASV}} = \frac{2\lambda_5}{\lambda_3 + \lambda_5}.$$

Their key discrepancy happens when discussing π_4. Under GPASV, $M_\omega(S_4) = \{5\}$, while under PASV, $\max(S_4) = \{3,5\}$, leading to the crucial difference between these two schemes.

From this observation, we emphasize following understandings:

- In terms of the support of the distribution of π, a GPASV under a cyclic graph may reduce to a PASV. However, their π distributions are still different;
- In PASV, at each stage t, the maximal set $\max(S_t)$ is the range of candidate players eligible for competing for π_t; however, in GPASV, this is completely different: the set of eligible players for π_t is **not** determined by $M_\omega(S_t)$, moreover, it cannot be determined locally, but only through the global rule $\pi \in \widetilde{\Pi}^{G_\omega}$.

E Proofs of Theoretical Results

E.1 Proof of Theorem 1

We first record the reduction from the four background axioms to the ROV class; this is the entry point for the GSCF/GWP/PVF analysis below.

Lemma 1 (Weber [42]). *A value ψ satisfies E + L + NP + M (Appendix B) if and only if $\psi = \nu^p$ for some probability distribution p on Π, where ν^p is defined in (2).*

A proof of Lemma 1 is given by Weber [42]; we take it as a starting point and focus on the GSCF/GWP/PVF analysis below.

Proof of Theorem 1. We prove the two implications $(\Rightarrow)$ and $(\Leftarrow)$ separately.

$(\Rightarrow)$ Suppose p is a ROV satisfying GSCF, GWP and PVF. By GSCF (6), there exist $s_\omega : 2^{[n]} \to \mathbb{R}_{\geq 0}$ and $c_{\lambda,\omega}(\cdot;\cdot)$ with $\sum_{i\in S} c_{\lambda,\omega}(i;S) = 1$ for every nonempty $S \subseteq [n]$ such that

$$p(\pi) \propto \prod_{t=1}^{n} s_\omega(S_t)\, c_{\lambda,\omega}(\pi_t; S_t), \quad \pi \in \Pi. \tag{13}$$

Fix any nonempty $S \subseteq [n]$ and any reference $j \in S$. For every $i \in S$, GWP (7) yields

$$c_{\lambda,\omega}(i;S) = c_{\lambda,\omega}(j;S) \cdot \frac{\lambda_i \exp\{-\beta \cdot V_\omega(i;S)\}}{\lambda_j \exp\{-\beta \cdot V_\omega(j;S)\}}.$$

Summing over $i \in S$ and using the GSCF normalization $\sum_{i\in S} c_{\lambda,\omega}(i;S) = 1$ identifies

$$c_{\lambda,\omega}(i;S) = \frac{\lambda_i \exp\{-\beta \cdot V_\omega(i;S)\}}{\sum_{k\in S} \lambda_k \exp\{-\beta \cdot V_\omega(k;S)\}}, \quad i \in S, \tag{14}$$

which determines the GSCF choice factor uniquely from (λ, ω, β).

Substituting (14) into (13) gives, for every $\pi \in \Pi$,

$$p(\pi) \propto \left(\prod_{t=1}^{n} \frac{s_\omega(S_t)}{\sum_{k\in S_t} \lambda_k \exp\{-\beta \cdot V_\omega(k;S_t)\}} \right) \cdot \prod_{t=1}^{n} \lambda_{\pi_t} \exp\{-\beta \cdot V_\omega(\pi_t; S_t)\}. \tag{15}$$

The exponents in the second product simplify by re-indexing the double sum

$$\sum_{t=1}^{n} V_\omega(\pi_t; S_t) = \sum_{t=1}^{n} \sum_{s=1}^{t} \omega_{\pi_t,\pi_s} = \sum_{1\leq s<t\leq n} \omega_{\pi_t,\pi_s} = V_\omega(\pi), \tag{16}$$

where the last equality re-indexes by the ordered pairs (π_s, π_t) for which π_s appears before π_t. Specializing to $\lambda_i \equiv 1$, (15) then reduces to

$$p(\pi) \propto \exp\{-\beta \cdot V_\omega(\pi)\} \cdot \prod_{t=1}^{n} \frac{s_\omega(S_t)}{\sum_{k\in S_t} \exp\{-\beta \cdot V_\omega(k;S_t)\}}.$$

Comparing with PVF, which asserts $p(\pi) \propto \exp\{-\beta \cdot V_\omega(\pi)\}$, the prefix product must be a π-independent constant: there exists $K > 0$ such that

$$\prod_{t=1}^{n} \frac{s_\omega(S_t)}{\sum_{k\in S_t} \exp\{-\beta \cdot V_\omega(k;S_t)\}} = K \quad \text{for every } \pi \in \Pi. \tag{17}$$

The factor on the left is independent of λ, so (17) continues to hold for arbitrary λ.

Returning to (15) for arbitrary λ, we factor each stage to expose the GPASV form:

$$
\begin{aligned}
p(\pi) &\propto \prod_{t=1}^{n} \frac{s_\omega(S_t)\,\lambda_{\pi_t}\exp\{-\beta\cdot V_\omega(\pi_t;S_t)\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot V_\omega(k;S_t)\}} \\
&= \left(\prod_{t=1}^{n}\frac{s_\omega(S_t)}{\sum_{k\in S_t}\exp\{-\beta\cdot V_\omega(k;S_t)\}}\right) \\
&\quad\times\prod_{t=1}^{n}\left[\frac{\lambda_{\pi_t}\exp\{-\beta\cdot V_\omega(\pi_t;S_t)\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot V_\omega(k;S_t)\}}\cdot\sum_{k\in S_t}\exp\{-\beta\cdot V_\omega(k;S_t)\}\right] \\
&\propto \prod_{t=1}^{n}\left[\frac{\lambda_{\pi_t}\exp\{-\beta\cdot V_\omega(\pi_t;S_t)\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot V_\omega(k;S_t)\}}\cdot\sum_{k\in S_t}\exp\{-\beta\cdot V_\omega(k;S_t)\}\right],
\end{aligned}
$$

which is exactly $p^{(\lambda,\omega)}(\pi)$ in (5).

($\Leftarrow$) Conversely, take $p = p^{(\lambda,\omega)}$ defined in (5). Setting

$$
s_\omega(S) := \sum_{k\in S}\exp\{-\beta\cdot V_\omega(k;S)\},\quad c_{\lambda,\omega}(i;S) := \frac{\lambda_i\exp\{-\beta\cdot V_\omega(i;S)\}}{\sum_{k\in S}\lambda_k\exp\{-\beta\cdot V_\omega(k;S)\}},
$$

we have $s_\omega : 2^{[n]} \to \mathbb{R}_{\geq 0}$, $\sum_{i\in S} c_{\lambda,\omega}(i;S) = 1$, and $p^{(\lambda,\omega)}(\pi) \propto \prod_t s_\omega(S_t)\,c_{\lambda,\omega}(\pi_t;S_t)$, so GSCF holds. The ratio $c_{\lambda,\omega}(i;S)/c_{\lambda,\omega}(j;S)$ matches (7) by construction, so GWP holds. When $\lambda_i \equiv 1$, the stage-wise product Part 1·Part 2 in (5) collapses to $\exp\{-\beta\cdot V_\omega(\pi_t;S_t)\}$, and (16) gives $\prod_t \exp\{-\beta\cdot V_\omega(\pi_t;S_t)\} = \exp\{-\beta\cdot V_\omega(\pi)\}$, recovering (4); hence PVF holds. □

E.2 Proof of Theorem 2

Proof. We will first isolate the dominant exponential factor in β that controls the limiting support, and then evaluate the residual factor on that support.

By (5), for every $\pi \in \Pi$,

$$
p^{(\lambda,\omega)}(\pi) \propto \prod_{t=1}^{n}\lambda_{\pi_t}\exp\{-\beta\cdot V_\omega(\pi_t;S_t)\}\cdot\frac{\sum_{k\in S_t}\exp\{-\beta\cdot V_\omega(k;S_t)\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot V_\omega(k;S_t)\}}.
$$

Using the identity (16), $\prod_{t=1}^{n}\exp\{-\beta\cdot V_\omega(\pi_t;S_t)\} = \exp\{-\beta\cdot V_\omega(\pi)\}$, so

$$
p^{(\lambda,\omega)}(\pi) \propto \exp\{-\beta\cdot V_\omega(\pi)\}\cdot A_\beta(\pi),\quad A_\beta(\pi) := \prod_{t=1}^{n}\lambda_{\pi_t}\cdot\frac{\sum_{k\in S_t}\exp\{-\beta\cdot V_\omega(k;S_t)\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot V_\omega(k;S_t)\}}. \tag{18}
$$

We now study the limit of $A_\beta(\pi)$ for a fixed $\pi \in \Pi$. For each stage t, set

$$
m_\omega(S_t) := \min_{k\in S_t} V_\omega(k;S_t),\quad M_\omega(S_t) = \arg\min_{k\in S_t} V_\omega(k;S_t),
$$

and factor out $e^{-\beta\cdot m_\omega(S_t)}$ from both the numerator and the denominator inside $A_\beta(\pi)$ to obtain

$$
\frac{\sum_{k\in S_t}\exp\{-\beta\cdot V_\omega(k;S_t)\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot V_\omega(k;S_t)\}} = \frac{\sum_{k\in S_t}\exp\{-\beta\cdot[V_\omega(k;S_t)-m_\omega(S_t)]\}}{\sum_{k\in S_t}\lambda_k\exp\{-\beta\cdot[V_\omega(k;S_t)-m_\omega(S_t)]\}}.
$$

Each exponent in the rewritten sums is nonpositive and equals zero precisely when $k \in M_\omega(S_t)$, so as $\beta\to\infty$ every term with $k \notin M_\omega(S_t)$ vanishes and the ratio tends to $|M_\omega(S_t)|/\sum_{k\in M_\omega(S_t)}\lambda_k \in (0,\infty)$. Taking the product over $t = 1,\dots,n$ yields

$$
A_\beta(\pi) \xrightarrow{\beta\to\infty} A_\infty(\pi) := \prod_{t=1}^{n}\frac{\lambda_{\pi_t}\,|M_\omega(S_t)|}{\sum_{k\in M_\omega(S_t)}\lambda_k} \in (0,\infty),\quad \pi\in\Pi. \tag{19}
$$

It remains to handle the exponential factor in (18). Recall $\widetilde{\Pi}^{G_\omega} = \arg\min_{\pi\in\Pi} V_\omega(\pi)$ and let

$$
V^\star := \min_{\pi\in\Pi} V_\omega(\pi).
$$

Multiplying every unnormalized weight in (18) by the common factor $e^{\beta \cdot V^\star}$ leaves the induced probability distribution unchanged, so

$$p^{(\lambda,\omega)}(\pi) \propto \exp\{-\beta \cdot [V_\omega(\pi) - V^\star]\} \cdot A_\beta(\pi), \quad \pi \in \Pi. \tag{20}$$

For $\pi \in \widetilde{\Pi}^{G_\omega}$, the exponential factor in (20) is identically 1, and by (19) the unnormalized weight converges to $A_\infty(\pi) \in (0, \infty)$. For $\pi \notin \widetilde{\Pi}^{G_\omega}$, we have $V_\omega(\pi) - V^\star > 0$ while $A_\beta(\pi)$ stays bounded by (19), so the unnormalized weight tends to 0. Consequently the total normalizing mass converges to the strictly positive quantity $\sum_{\pi \in \widetilde{\Pi}^{G_\omega}} A_\infty(\pi)$, and

$$\lim_{\beta \to \infty} p^{(\lambda,\omega)}(\pi) = \begin{cases} \dfrac{A_\infty(\pi)}{\sum_{\pi' \in \widetilde{\Pi}^{G_\omega}} A_\infty(\pi')}, & \pi \in \widetilde{\Pi}^{G_\omega}, \\ 0, & \pi \notin \widetilde{\Pi}^{G_\omega}, \end{cases}$$

which, by the definition of A_∞ in (19), is exactly (8). □

E.3 Proof of Proposition 1

Proof. Recall $S_t = \{\pi_1, \ldots, \pi_t\}$, so the prefix shared by π and π' before the swapped pair is $S_{i-1} = \{\pi_1, \ldots, \pi_{i-1}\}$, and

$$S_i = S_{i-1} \cup \{a\}, \quad S'_i = S_{i-1} \cup \{b\}, \quad S_{i+1} = S'_{i+1} = S_{i-1} \cup \{a, b\}.$$

For any stage $t \notin \{i, i+1\}$, swapping positions i and $i+1$ leaves both π_t and S_t unchanged, so the corresponding factor in (5) is identical for π and π' and cancels in the ratio $p^{(\lambda,\omega)}(\pi')/p^{(\lambda,\omega)}(\pi)$. It therefore suffices to compare the contributions at stages $t = i$ and $t = i+1$.

Write the stagewise factor of (5) at (x, S) as

$$F(x; S) := \frac{\lambda_x \exp\{-\beta \cdot V_\omega(x; S)\}}{\sum_{k \in S} \lambda_k \exp\{-\beta \cdot V_\omega(k; S)\}} \cdot \sum_{k \in S} \exp\{-\beta \cdot V_\omega(k; S)\} = \frac{\lambda_x \exp\{-\beta \cdot V_\omega(x; S)\}}{\zeta_{\lambda,\omega}(S)}, \tag{21}$$

where the second equality follows from the definition of $\zeta_{\lambda,\omega}(S)$ in the proposition. With this notation,

$$\frac{p^{(\lambda,\omega)}(\pi')}{p^{(\lambda,\omega)}(\pi)} = \frac{F(b; S'_i)\, F(a; S'_{i+1})}{F(a; S_i)\, F(b; S_{i+1})}. \tag{22}$$

Since $S_{i+1} = S'_{i+1}$, the factor $\zeta_{\lambda,\omega}(S_{i+1})$ appears once in the numerator and once in the denominator of (22) and cancels. The λ factors λ_a and λ_b likewise appear once in each, and cancel. Substituting (21) and collecting the surviving terms yields

$$\frac{p^{(\lambda,\omega)}(\pi')}{p^{(\lambda,\omega)}(\pi)} = \exp\Big\{ -\beta \big[V_\omega(b; S'_i) + V_\omega(a; S_{i+1}) - V_\omega(a; S_i) - V_\omega(b; S_{i+1})\big] \Big\} \cdot \frac{\zeta_{\lambda,\omega}(S_i)}{\zeta_{\lambda,\omega}(S'_i)}. \tag{23}$$

It remains to evaluate the bracketed exponent. Using $V_\omega(x; S) = \sum_{j \in S} \omega_{xj}$ and the convention $\omega_{xx} = 0$,

$$V_\omega(b; S'_i) = \sum_{j \in S_{i-1}} \omega_{bj}, \quad V_\omega(a; S_{i+1}) = \sum_{j \in S_{i-1}} \omega_{aj} + \omega_{ab},$$
$$V_\omega(a; S_i) = \sum_{j \in S_{i-1}} \omega_{aj}, \quad V_\omega(b; S_{i+1}) = \sum_{j \in S_{i-1}} \omega_{bj} + \omega_{ba}.$$

Hence

$$V_\omega(b; S'_i) + V_\omega(a; S_{i+1}) - V_\omega(a; S_i) - V_\omega(b; S_{i+1}) = \omega_{ab} - \omega_{ba}.$$

Substituting into (23) gives (36). □

E.4 Proof of Proposition 2

Proof. Throughout we work in the multiplicative representation $\widetilde{\omega}_{ij} = \exp(-\beta \omega_{ij})$ and $\widetilde{\omega}^S_k = \prod_{j \in S} \widetilde{\omega}_{kj}$ of Appendix C, and abbreviate

$$\mu_{\lambda,\omega}(S) := \sum_{k \in S} \lambda_k \widetilde{\omega}^S_k, \qquad s_\omega(S) := \sum_{k \in S} \widetilde{\omega}^S_k.$$

Under the Scenario 1 total order, $\widetilde{\omega}_{ij} = \widetilde{\omega}_0$ for $i < j$ and $\widetilde{\omega}_{ij} = 1$ otherwise (with the convention $\widetilde{\omega}_{ii} = 1$), so for every $k \in [n]$ and $S \subseteq [n]$,

$$\widetilde{\omega}_k^S = \widetilde{\omega}_0^{|\{j \in S:\, j > k\}|}. \tag{24}$$

By (44) and linearity of the ROV,

$$\psi_i(U) = \sum_{j:\, i \in T_j} c_j \cdot \mathbb{P}_{\pi \sim p^{(\lambda,\omega)}}\big(\, i \text{ is the last member of } T_j \text{ in } \pi \,\big), \tag{25}$$

so it suffices to fix one interval $T = T_j = \{\ell, \dots, r\}$ (writing $\ell = \ell_j$, $r = r_j$) and prove

$$\mathbb{P}_{\pi \sim p^{(\lambda,\omega)}}\big(\, i \text{ is the last of } T \text{ in } \pi \,\big) = \frac{\lambda_i \widetilde{\omega}_0^{\,r-i}}{\sum_{k=\ell}^{r} \lambda_k \widetilde{\omega}_0^{\,r-k}}, \qquad i \in T. \tag{26}$$

We first observe that the GPASV distribution admits a particularly clean form under Scenario 1. By (24), for any $S \subseteq [n]$ with $|S| = t$,

$$s_\omega(S) = \sum_{k \in S} \widetilde{\omega}_0^{|\{j \in S:\, j > k\}|} = \sum_{q=0}^{t-1} \widetilde{\omega}_0^{\,q},$$

because the exponent $|\{j \in S : j > k\}|$ ranges over $\{0, 1, \dots, t-1\}$ as k ranges over S, regardless of which elements of $[n]$ populate S. Hence $s_\omega(S_t)$ depends on π only through $|S_t| = t$, and $\prod_{t=1}^{n} s_\omega(S_t)$ is a π-independent constant. Substituting into (5) (equivalently (11)) yields

$$p^{(\lambda,\omega)}(\pi) \;\propto\; \prod_{t=1}^{n} \frac{\lambda_{\pi_t} \widetilde{\omega}_{\pi_t}^{S_t}}{\mu_{\lambda,\omega}(S_t)}, \tag{27}$$

which is exactly the joint law of the following *backward sequential sampler*: set $S_n = [n]$, and for $t = n, n-1, \dots, 1$, draw

$$\mathbb{P}(\pi_t = k \mid S_t) = \frac{\lambda_k \widetilde{\omega}_k^{S_t}}{\mu_{\lambda,\omega}(S_t)}, \qquad k \in S_t, \tag{28}$$

then set $S_{t-1} := S_t \setminus \{\pi_t\}$.

Under this sampler, the last member of T to appear in π (in the forward ordering) is the *first* T-element drawn in reverse time. Define the reverse-time stopping stage

$$\tau := \max\{t \in [n] :\, \pi_t \in T\}.$$

At stage τ no T-element has yet been drawn, so $T \subseteq S_\tau$, and (28) gives

$$\mathbb{P}(\pi_\tau = i \mid S_\tau,\, \pi_\tau \in T) = \frac{\lambda_i \widetilde{\omega}_i^{S_\tau}}{\sum_{k \in T} \lambda_k \widetilde{\omega}_k^{S_\tau}}, \qquad i \in T. \tag{29}$$

By (24), for $k \in T = \{\ell, \dots, r\}$,

$$|\{j \in S_\tau : j > k\}| = |\{j \in T : j > k\}| + |\{j \in S_\tau \setminus T : j > k\}| = (r-k) + |\{j \in S_\tau \setminus T : j > k\}|.$$

Since $[n] \setminus T = \{1, \dots, \ell-1\} \cup \{r+1, \dots, n\}$ and $k \in [\ell, r]$, the set $\{j \in [n] \setminus T : j > k\}$ equals $\{r+1, \dots, n\}$, which is independent of the choice of $k \in T$. Writing

$$C(S_\tau) := \widetilde{\omega}_0^{|S_\tau \cap \{r+1, \dots, n\}|},$$

we therefore have $\widetilde{\omega}_k^{S_\tau} = \widetilde{\omega}_0^{\,r-k} \cdot C(S_\tau)$ for every $k \in T$, and the factor $C(S_\tau)$ cancels between numerator and denominator of (29):

$$\mathbb{P}(\pi_\tau = i \mid S_\tau,\, \pi_\tau \in T) = \frac{\lambda_i \widetilde{\omega}_0^{\,r-i}}{\sum_{k=\ell}^{r} \lambda_k \widetilde{\omega}_0^{\,r-k}}.$$

The right-hand side does not depend on S_τ, and τ is well-defined almost surely (some T-element is eventually drawn), so marginalization yields (26). Substituting into (25) completes the proof. □

E.5 Proof of Proposition 3

Proof. We retain the notation $\widetilde{\omega}_k^S$, $s_\omega(S)$, and $\mu_{\lambda,\omega}(S)$ from the proof of Proposition 2. Specializing (44) to $T_j = B_j$ and using linearity of the ROV, for $i \in B_j$,

$$\psi_i(U) = c_j \cdot \mathbb{P}_{\pi\sim p^{(\lambda,\omega)}}\big(\, i \text{ is the last of } B_j \text{ in } \pi \,\big), \tag{30}$$

so it suffices to show that

$$\mathbb{P}_{\pi\sim p^{(\lambda,\omega)}}\big(\, i \text{ is the last of } B_j \text{ in } \pi \,\big) = \frac{1}{|B_j|}, \qquad i \in B_j. \tag{31}$$

We will prove (31) by exhibiting a bijection of $[n]$ that leaves the GPASV distribution invariant and acts as a single cycle on B_j. Let $B_j = \{b_1, b_2, \dots, b_m\}$ with $m = |B_j|$, labeled so that the within-block directed cycle is $b_1 \to b_2 \to \cdots \to b_m \to b_1$, and adopt the cyclic convention $b_{m+1} := b_1$ throughout this proof. Thus $\widetilde{\omega}_{b_s\, b_{s+1}} = \widetilde{\omega}_j^{\mathrm{cyc}}$ for $s = 1, \dots, m$ (the common within-block cycle weight), while every other ordered within-block pair is a non-edge with $\widetilde{\omega}$-value 1. Define the bijection $\phi : [n] \to [n]$ by

$$\phi(b_s) = b_{s+1}\ (s = 1, \dots, m), \qquad \phi(x) = x \text{ for } x \in [n] \setminus B_j.$$

We first establish the pairwise invariance

$$\widetilde{\omega}_{\phi(k)\phi(\ell)} = \widetilde{\omega}_{k\ell}, \qquad k, \ell \in [n]. \tag{32}$$

If $k, \ell \in B_j$, the shift $s \mapsto s+1$ preserves the cycle edge set $\{(b_s, b_{s+1}) : s = 1, \dots, m\}$ (using the convention $b_{m+1} = b_1$) and its common weight $\widetilde{\omega}_j^{\mathrm{cyc}}$, so both sides of (32) agree (either both equal $\widetilde{\omega}_j^{\mathrm{cyc}}$ or both equal 1). If $k \in B_j$ and $\ell \in B_{m'}$ with $B_{m'} \neq B_j$, then by assumption $\widetilde{\omega}_{k\ell}$ is constant in $(k, \ell) \in B_j \times B_{m'}$ (the block-pair common weight if $B_j \to B_{m'}$ is a block edge, otherwise 1), and $\phi(\ell) = \ell$, hence $\widetilde{\omega}_{\phi(k)\phi(\ell)} = \widetilde{\omega}_{\phi(k)\ell} = \widetilde{\omega}_{k\ell}$; the case $k \notin B_j, \ell \in B_j$ is symmetric, and the case $k, \ell \notin B_j$ is trivial since ϕ fixes both arguments. Moreover $\lambda_{\phi(k)} = \lambda_k$ for all $k \in [n]$, since λ is constant on B_j and ϕ fixes every element of $[n] \setminus B_j$.

From (32) we obtain, for every $k \in [n]$ and $S \subseteq [n]$,

$$\widetilde{\omega}_{\phi(k)}^{\phi(S)} = \prod_{\ell \in S} \widetilde{\omega}_{\phi(k)\phi(\ell)} = \prod_{\ell \in S} \widetilde{\omega}_{k\ell} = \widetilde{\omega}_k^S, \tag{33}$$

where the first equality uses the reindexing $\ell \mapsto \phi(\ell)$ of the product over $\phi(S)$. Consequently $s_\omega(\phi(S)) = s_\omega(S)$ and, since $\lambda_{\phi(k)} = \lambda_k$, $\mu_{\lambda,\omega}(\phi(S)) = \mu_{\lambda,\omega}(S)$.

For any permutation $\pi = (\pi_1, \dots, \pi_n) \in \Pi$, write $\phi(\pi) := (\phi(\pi_1), \dots, \phi(\pi_n))$ for the component-wise action of ϕ on the tuple π; since ϕ is a bijection on $[n]$, $\phi(\pi) \in \Pi$ as well. Denote its prefix sets by $\pi' := \phi(\pi)$, so that $\pi'_t = \phi(\pi_t)$ and $S_t^{\pi'} = \phi(S_t^\pi)$. Combining the three invariances above, each stagewise factor of (5) is preserved under $\pi \mapsto \pi'$:

$$\frac{\lambda_{\pi'_t} \widetilde{\omega}_{\pi'_t}^{S_t^{\pi'}}}{\mu_{\lambda,\omega}(S_t^{\pi'})} \cdot s_\omega(S_t^{\pi'}) = \frac{\lambda_{\pi_t} \widetilde{\omega}_{\pi_t}^{S_t^{\pi}}}{\mu_{\lambda,\omega}(S_t^{\pi})} \cdot s_\omega(S_t^{\pi}).$$

Taking the product over $t = 1, \dots, n$ yields

$$p^{(\lambda,\omega)}(\phi(\pi)) = p^{(\lambda,\omega)}(\pi), \qquad \pi \in \Pi. \tag{34}$$

It remains to convert (34) into uniformity of the last-of-B_j element. Define $t^\star(\pi) := \max\{t \in [n] : \pi_t \in B_j\}$ and $L(\pi) := \pi_{t^\star(\pi)}$. Because $\phi(B_j) = B_j$, for $\pi' = \phi(\pi)$ we have $\{t : \pi'_t \in B_j\} = \{t : \pi_t \in B_j\}$, so $t^\star(\pi') = t^\star(\pi)$ and

$$L(\phi(\pi)) = \pi'_{t^\star(\pi)} = \phi(\pi_{t^\star(\pi)}) = \phi(L(\pi)).$$

Combining this equivariance with (34) and bijectivity of ϕ, for every $i \in B_j$,

$$\mathbb{P}\big(L(\pi) = i\big) = \mathbb{P}\big(L(\phi(\pi)) = \phi(i)\big) = \mathbb{P}\big(L(\pi) = \phi(i)\big).$$

Since ϕ acts on B_j as a single m-cycle, iterating this identity shows that $\mathbb{P}(L(\pi) = i)$ is constant in $i \in B_j$, so it equals $1/|B_j|$. This proves (31), and substitution into (30) gives $\psi_i(U) = c_j/|B_j|$ for $i \in B_j$. □

F Computational Details for GPASV

Section 4 outlines the high-level computational pipeline. This appendix collects the detailed derivations and algorithms.

F.1 Permutation Sampling

GPASV is defined as an expectation under $p^{(\lambda,\omega)}$, so the principal computational primitive is sampling permutations from this distribution. We describe an adjacent-swap Metropolis–Hastings sampler with a local acceptance ratio (Section F.1.1), and a greedy initialization that mirrors the GPASV choice factor (Section F.1.2).

F.1.1 Adjacent-Swap Metropolis–Hastings Sampler

GPASV is defined as an expectation with respect to a permutation distribution on the full space Π. Accordingly, the main computational task is to sample permutations approximately from $p^{(\lambda,\omega)}$. Adjacent-swap Metropolis-Hastings (MH) is a canonical local approach for sampling permutations, both in uniform and non-uniform settings [22, 4, 26]. We adopt this proposal because it preserves the permutation structure while allowing the target distribution to be the GPASV distribution on the full space Π.

Let p be any target distribution on Π and let $q(\pi, \pi')$ be a proposal kernel. The general MH acceptance probability is

$$A_{\mathrm{MH}}(\pi, \pi') := \min\left\{1, \frac{p(\pi')q(\pi', \pi)}{p(\pi)q(\pi, \pi')}\right\}. \tag{35}$$

In our implementation, given the current permutation $\pi = (\pi_1, \ldots, \pi_n)$, we choose an index $i \in [n-1]$ uniformly at random and propose π' by swapping the adjacent pair $a := \pi_i$, $b := \pi_{i+1}$. This adjacent-swap proposal is symmetric, so $q(\pi, \pi') = q(\pi', \pi)$ and the proposal term in (35) cancels. Therefore the acceptance probability reduces to

$$A_{\mathrm{MH}}(\pi, \pi') = \min\left\{1, \frac{p^{(\lambda,\omega)}(\pi')}{p^{(\lambda,\omega)}(\pi)}\right\}.$$

Proposition 1 (Local Adjacent-Swap Ratio for GPASV)**.** *Let $\pi' \in \Pi$ be obtained from $\pi \in \Pi$ by swapping the adjacent pair $(a, b) = (\pi_i, \pi_{i+1})$ at positions $(i, i+1)$. Write*

$$S_i = \{\pi_1, \ldots, \pi_{i-1}, a\}, \quad S_i' = \{\pi_1, \ldots, \pi_{i-1}, b\},$$

and define, for any nonempty $S \subseteq [n]$,

$$\zeta_{\lambda,\omega}(S) := \frac{\sum_{k\in S} \lambda_k \exp\{-\beta \cdot V_\omega(k; S)\}}{\sum_{k\in S} \exp\{-\beta \cdot V_\omega(k; S)\}}.$$

Then

$$\frac{p^{(\lambda,\omega)}(\pi')}{p^{(\lambda,\omega)}(\pi)} = \exp\{-\beta(\omega_{ab} - \omega_{ba})\} \cdot \frac{\zeta_{\lambda,\omega}(S_i)}{\zeta_{\lambda,\omega}(S_i')}. \tag{36}$$

Proposition 1 shows that the acceptance ratio depends only on the swapped pair (a, b) and the two competing sets S_i, S_i', not on the entire permutation. This locality is the key structural reason why adjacent-swap MH remains practical for GPASV.

Algorithm 1 summarizes the resulting sampler.

F.1.2 Greedy Initialization

Algorithm 1 is valid for any initial permutation $\pi^{(0)} \in \Pi$, but in practice, initialization can matter because GPASV lives on the full permutation space Π while the adjacent-swap proposal is purely local. This is especially relevant in concentrated regimes, where a poor starting permutation can increase the transient cost before the chain reaches a representative high-probability region.

In a DAG, this issue is comparatively mild. A zero-violation initialization is easy to obtain, since finding one linear extension is computationally feasible (e.g. Kahn's algorithm [19]). By contrast,

Algorithm 1 Adjacent-swap MH for sampling from $p^{(\lambda,\omega)}$

Require: Parameters (λ, ω), lazy probability $\xi \in [0, 1)$, burn-in B, thinning τ, target sample size N_{MC}

Initialize $\pi^{(0)} \in \Pi$ via Algorithm 2
Initialize the sample set $\mathcal{T} \leftarrow \emptyset$
for $t = 1$ to $B + \tau(N_{\mathrm{MC}} - 1) + 1$ **do**
  Set $\pi^{(t)} \leftarrow \pi^{(t-1)}$
  with probability $1 - \xi$ **do**
    Draw i uniformly from $[n - 1]$
    Let $a \leftarrow \pi_i^{(t-1)}$ and $b \leftarrow \pi_{i+1}^{(t-1)}$
    Form π' by swapping the adjacent pair (a, b)
    Compute the local ratio in (36)
    Accept π' with probability $\min\{1, p^{(\lambda,\omega)}(\pi')/p^{(\lambda,\omega)}(\pi^{(t-1)})\}$
    if accepted **then**
      $\pi^{(t)} \leftarrow \pi'$
    end if
  end with
  if $t > B$ and $(t - B - 1) \bmod \tau = 0$ **then**
    Append $\pi^{(t)}$ to $\mathcal{T}$
  end if
end for
return $\mathcal{T}$

Algorithm 2 Greedy initialization for GPASV

Require: Parameters (λ, ω)

Initialize the remaining set $R \leftarrow [n]$
for $t = n$ down to 1 **do**
  For each $i \in R$, compute the weight $w(i; R) \leftarrow \lambda_i \exp\{-\beta \cdot V_\omega(i; R)\}$
  Sample $i^\star \in R$ with probability proportional to $w(i; R)$
  Set $\pi_t^{(0)} \leftarrow i^\star$
  Update $R \leftarrow R \setminus \{i^\star\}$
end for
return $\pi^{(0)}$

on a general directed graph a linear extension may not exist at all; a good starting permutation must instead balance unavoidable edge violations against soft priorities. Rather than solving a separate global optimization problem, we use a stagewise greedy initialization that mirrors the GPASV choice factor itself.

For a current remaining set $R \subseteq [n]$ and a player $i \in R$, the quantity $\exp\{-\beta \cdot V_\omega(i; R)\} = \exp\{-\beta \sum_{j \in R} \omega_{ij}\}$ is exactly the stage-wise edge factor incurred when i is placed at the current rightmost position relative to the other players still in R. Larger values of this factor (equivalently, smaller $V_\omega(i; R)$) correspond to fewer or weaker immediate priority violations, while neutral non-edges ($\omega_{ij} = 0$) contribute factor 1. The strategy constructs the permutation backward. At each step it samples a player with probability proportional to the local GPASV choice factor $\lambda_i \exp\{-\beta \cdot V_\omega(i; R)\}$, places that player at the rightmost unfilled position, and removes it from the remaining set.

Algorithm 2 summarizes the resulting initialization rule.

When G_ω is a DAG, every maximal node in the induced subgraph on R has $V_\omega(i; R) = 0$, whereas every non-maximal node has $V_\omega(i; R) > 0$. Thus Algorithm 2 systematically favors maximal nodes (whose stage-wise factor equals 1) while still retaining the node-weight term λ_i in the local sampling weights. In the hard-priority limit $\beta \to \infty$, non-maximal nodes receive vanishing relative weight, so the rule collapses to a λ-weighted randomized linear extension.

F.2 GPASV Estimators

We use two estimators for the GPASV target in (2): a direct permutation-based Monte Carlo estimator and a subset-reweighted surrogate estimator.

F.2.1 Direct Permutation Monte Carlo

The most direct estimator follows from the random order value representation in (2). A natural choice is to sample permutations $\pi^{(1)}, \dots, \pi^{(N_{\mathrm{MC}})}$ from $p^{(\lambda,\omega)}$ and average the corresponding marginal contributions along those permutations. This mirrors the standard permutation-based estimator for the classical Shapley value, where the order distribution is uniform over Π [6]. Define the marginal contribution

$$\Delta_i(\pi; U) := U(\pi^i \cup \{i\}) - U(\pi^i).$$

Then, if $\pi^{(1)}, \dots, \pi^{(N_{\mathrm{MC}})} \sim p^{(\lambda,\omega)}$, the direct estimator is

$$\widehat{\psi}_i^{\mathrm{MC}}(U) := \frac{1}{N_{\mathrm{MC}}} \sum_{m=1}^{N_{\mathrm{MC}}} \Delta_i(\pi^{(m)}; U). \tag{37}$$

Its finite-sample behavior is controlled by the quality and effective size of the sampled permutation pool.

F.2.2 Surrogate-Assisted Subset Estimator

The second route provides a bridge between GPASV computation and a line of semivalue estimation methods familiar in data valuation and feature attribution. Semivalues are a class of attribution rules that generalize the Shapley value by relaxing the efficiency (E in Appendix B) [10]: fixing n, a semivalue with weights $a_0, \dots, a_{n-1}$ satisfying $a_s \geq 0$ and $\sum_{s=0}^{n-1} \binom{n-1}{s} a_s = 1$ is defined by $\varphi_i(U) = \sum_{S \subseteq [n] \setminus \{i\}} a_{|S|} \big\{U(S \cup \{i\}) - U(S)\big\}$. Unlike a random order value, semivalues average subset-based marginal contributions rather than marginal contributions along sampled permutations.

Existing semivalue estimators differ in their observation design and in the rule used to recover the target from sampled utilities; these include regression-based methods [32, 15, 14], direct weighted-average methods [28, 40], and surrogate-adjusted residual estimators [44]. A recent work [30] shows that these can be described by a unified framework. We adapt [30] and extend its method to the setting where the priority relationship between data points should be considered.

The main obstacle is that GPASV is not a semivalue in general: random order values including GPASV may break symmetry, whereas semivalues need not satisfy efficiency. Nevertheless, the subset-linear viewpoint remains useful. Following [28, 30], we first rewrite a semivalue as a pure weighted sum over subsets:

$$\varphi_i(U) = \sum_{S \subseteq [n]} \rho_i(S) U(S), \quad \rho_i(S) = \mathbb{1}\{i \in S\} \rho_i^+(S) - \mathbb{1}\{i \notin S\} \rho_i^-(S),$$

with $\rho_i^+(S) = a_{|S|-1}$, $\rho_i^-(S) = a_{|S|}$. The signed coefficient $\rho_i(S)$ depends on whether i belongs to S, but the positive and negative subset weights $\rho_i^+(S)$ and $\rho_i^-(S)$ are anonymous functions of coalition size, and known in closed form for several semivalues including the (beta) Shapley value [38, 25] and the (weighted) Banzhaf value [1, 27].

Random order values admit an analogous subset-based representation, but the positive and negative coefficients depend on the player i and the actual subset S. If a random order value is induced by a distribution p on Π,

$$\rho_i^+(S) = \mathbb{P}_{\pi \sim p}\big(\pi^i = S \setminus \{i\}\big), \quad \rho_i^-(S) = \mathbb{P}_{\pi \sim p}\big(\pi^i = S\big).$$

For GPASV, $p = p^{(\lambda,\omega)}$, so these coefficients depend on the player, the actual subset, and the priority structure encoded by (λ, ω); they are not cardinality-based closed-form weights. To make the semivalue-style construction usable, we first draw and store utility-free permutations $\widetilde{\pi}^{(1)}, \dots, \widetilde{\pi}^{(M)} \sim p^{(\lambda,\omega)}$ and estimate these coefficients by

$$\widehat{\rho}_i(S) := \frac{1}{M} \sum_{r=1}^{M} \Big[\mathbb{1}\{i \in S\} \mathbb{1}\{(\widetilde{\pi}^{(r)})^i = S \setminus \{i\}\} - \mathbb{1}\{i \notin S\} \mathbb{1}\{(\widetilde{\pi}^{(r)})^i = S\}\Big]. \tag{38}$$

To build a shared proposal over subsets, we use

$$\widehat{A}(S) := \sum_{i=1}^{n} |\widehat{\rho}_i(S)|, \quad \widehat{\mathcal{S}}_\rho := \{S \subseteq [n] : \widehat{A}(S) > 0\}, \quad \widehat{q}(S) := \frac{\widehat{A}(S)}{\sum_{T \in \widehat{\mathcal{S}}_\rho} \widehat{A}(T)}, \quad S \in \widehat{\mathcal{S}}_\rho.$$

This allocates more mass to subsets that carry larger aggregate coefficient magnitude across players. Assume the proposal $\widehat{q}$ satisfies $\widehat{q}(S) > 0$ whenever $\rho_i(S) \neq 0$ for some i. With $\gamma_i(S) := \rho_i(S)/\widehat{q}(S)$ and $\widehat{\gamma}_i(S) := \widehat{\rho}_i(S)/\widehat{q}(S)$, GPASV admits the rewritten form

$$\psi_i(U) = \mathbb{E}_{S \sim \widehat{q}}\left[\gamma_i(S) U(S)\right].$$

Since random order values still obey linearity, for any fixed surrogate h, $\psi_i(U) = \psi_i(h) + \psi_i(U - h)$. Hence, conditional on a fixed proposal $\widehat{q}$ with valid support and using exact ρ_i, the estimator

$$\psi_i(h) + \frac{1}{K_{\text{adjust}}} \sum_{k=1}^{K_{\text{adjust}}} \gamma_i(S^{(k)})\big(U(S^{(k)}) - h(S^{(k)})\big), \quad S^{(k)} \sim \widehat{q},$$

is unbiased for $\psi_i(U)$, where K_{adjust} is the number of residual-correction subsets. In practice we use the plug-in form below with $\widehat{\rho}_i$:

$$\widehat{\psi}_i^{\text{2stage}} := \widehat{\psi}_i(\widehat{h}) + \frac{1}{K_{\text{adjust}}} \sum_{k=1}^{K_{\text{adjust}}} \widehat{\gamma}_i(S^{(k)})\Big(U(S^{(k)}) - \widehat{h}(S^{(k)})\Big), \quad S^{(k)} \sim \widehat{q}. \tag{39}$$

It remains to specify how $\widehat{h}$ is obtained. We fit the surrogate on training subsets sampled independently from $\widehat{q}$, separately from the subsets later used for residual correction. Given K_{train} evaluated training subsets $T^{(1)}, \ldots, T^{(K_{\text{train}})} \sim \widehat{q}$ and a surrogate class $\mathcal{H}$, we adapt the weighted least squares procedure in [30] to the GPASV setting by defining

$$\widehat{h} \in \arg\min_{h \in \mathcal{H}} \sum_{\ell=1}^{K_{\text{train}}} \widehat{W}(T^{(\ell)}) \left\{U(T^{(\ell)}) - h(T^{(\ell)})\right\}^2, \quad \widehat{W}(S) := \sum_{i=1}^{n} \frac{\widehat{\rho}_i(S)^2}{\widehat{q}(S)^2}.$$

The weight $\widehat{W}(S)$ emphasizes subsets that contribute more strongly to the vector of GPASV coordinates.

For a linear surrogate [32] with fitted form $\widehat{h}_{\text{lin}}(S) = U(\emptyset) + \sum_{k=1}^{n} \widehat{a}_k \, \mathbb{1}\{k \in S\}$, we have $\widehat{\psi}_i(\widehat{h}_{\text{lin}}) = \widehat{a}_i$. For a quadratic surrogate [15] with selected interaction set $\mathcal{I} \subseteq \{\{k, \ell\} : 1 \leq k < \ell \leq n\}$,

$$\widehat{h}_{\text{quad}}(S) = U(\emptyset) + \sum_{k=1}^{n} \widehat{a}_k \, \mathbb{1}\{k \in S\} + \sum_{\{k,\ell\} \in \mathcal{I}} \widehat{b}_{k\ell} \, \mathbb{1}\{k, \ell \in S\},$$

where $\widehat{b}_{k\ell} = \widehat{b}_{\ell k}$ and $\widehat{b}_{ij} = 0$ if $\{i, j\} \notin \mathcal{I}$. For brevity in the calculations below, let $\pi_i^{-1} \in [n]$ denote the position of player i in π, i.e., $\pi_i^{-1} = t$ iff $\pi_t = i$; the prefix set π^i from (2) can then be written as $\pi^i = \{j \in [n] : \pi_j^{-1} < \pi_i^{-1}\}$. With this notation, the one-step marginal term satisfies $\Delta_i(\pi; \widehat{h}_{\text{quad}}) = \widehat{a}_i + \sum_{j \neq i} \widehat{b}_{ij} \, \mathbb{1}\{\pi_j^{-1} < \pi_i^{-1}\}$. Taking expectation over $\pi \sim p^{(\lambda,\omega)}$ yields

$$\widehat{\psi}_i(\widehat{h}_{\text{quad}}) = \widehat{a}_i + \sum_{j \neq i} \eta_{ji} \widehat{b}_{ij}, \quad \eta_{ji} := \mathbb{P}_{\pi \sim p^{(\lambda,\omega)}}\left(\pi_j^{-1} < \pi_i^{-1}\right).$$

In practice these pairwise probabilities are estimated from previously stored utility-free permutations:

$$\widehat{\eta}_{ji} := \frac{1}{M} \sum_{r=1}^{M} \mathbb{1}\left\{(\widetilde{\pi}^{(r)})_j^{-1} < (\widetilde{\pi}^{(r)})_i^{-1}\right\}.$$

We use a fixed training/correction split for simplicity; optimizing this allocation for the two-stage estimator is orthogonal to GPASV and left to future work.

F.3 Acceleration via Computational Reuse

The sampler above addresses the distributional side of GPASV, but in many applications the dominant cost is not the MH move itself. Rather, it is often the repeated evaluation of the cooperative game. This imbalance is well known in the broader Shapley literature: in feature attribution, utility evaluation may require many model inferences [32]; in data valuation, it may require repeated model retraining [16, 40]. This motivates two complementary acceleration mechanisms: reusing utility evaluations on repeated subsets, and reusing permutations across nearby target distributions.

F.3.1 Utility Reuse Across Repeated Subsets

The utility-caching mechanism described here applies to both estimators in Sections F.2.1 and F.2.2. Given sampled permutations $\pi^{(1)}, \dots, \pi^{(N_{\mathrm{MC}})}$, estimating GPASV requires the marginal contributions $\Delta_i(\pi^{(m)}; U)$ from (37):

$$\Delta_i(\pi^{(m)}; U), \quad i \in [n],\ m \in [N_{\mathrm{MC}}],$$

Each term is determined by a subset induced by the relative order in $\pi^{(m)}$. Because adjacent-swap proposals modify permutations only locally, and because many sampled permutations share long initial segments, the same subset often appears repeatedly across MCMC iterations and across players.

We therefore maintain a cache of utility evaluations indexed by the subset itself. Whenever a subset S is encountered for the first time, we evaluate $U(S)$ once and store the result; every later appearance of the same subset reuses the cached value. The same cache applies whether S arises as a permutation prefix or as a subset sampled from $\widehat{q}$ in the surrogate estimator. This does not change the estimator at all, but it can reduce the number of expensive utility calls by a large factor.

The practical consequence is that the total runtime separates naturally into two parts:

$$\text{runtime} \approx (\#\text{ unique subsets}) \times (\text{cost per utility evaluation}) \ + \ (\text{sampling overhead}).$$

This decomposition is particularly important in applications such as LLM evaluation, where utility computation can be much more expensive than sampling permutations.

F.3.2 SNIS for Permutation Reuse Across Priority Sweeps

The second acceleration mechanism is useful when the target distribution changes but the underlying attribution problem remains fixed. A canonical case is priority sweeping. Section 3.5 sweeps a single λ_i over $(0, \infty)$ to probe the sensitivity of one player's valuation. A useful complementary view, when the relative scales among players are themselves meaningful, is to fix a *latent* soft priority $\widetilde{\lambda}$ (with $\widetilde{\lambda}_i \geq 0$) and control its overall strength through a scalar temperature $\alpha \geq 0$ via

$$\lambda_i := \exp\{-\alpha \widetilde{\lambda}_i\}. \tag{40}$$

α is its λ-side analogue of β: $\alpha = 0$ removes the soft-priority contrast (all $\lambda_i \equiv 1$), while $\alpha \to \infty$ pushes players with larger $\widetilde{\lambda}_i$ progressively earlier in π. Substituting (40) into (5) yields a two-parameter family

$$p^{(\lambda,\omega)}_{\alpha,\beta}(\pi) := p^{(\lambda,\omega)}(\pi)\Big|_{\lambda_i = \exp\{-\alpha\widetilde{\lambda}_i\}},$$

and a sweep traces a continuous curve in (α, β)-space rather than along a single coordinate. This is the parameterization used, e.g., in our LLM application (Section 6).

Within such a sweep, instead of redrawing a full Monte Carlo sample at every new (α, β), we may reuse previously sampled permutations through importance weighting. Let

$$p(\pi) := p^{(\lambda,\omega)}_{\alpha,\beta}(\pi), \quad p'(\pi) := p^{(\lambda,\omega)}_{\alpha',\beta'}(\pi),$$

and let $\widetilde{p}$ and $\widetilde{p}'$ denote the corresponding unnormalized PMFs. Equivalently, we may write

$$p(\pi) = \frac{\widetilde{p}(\pi)}{Z_p}, \quad p'(\pi) = \frac{\widetilde{p}'(\pi)}{Z_{p'}},$$

where the normalizing constants Z_p and $Z_{p'}$ are intractable in general. Thus, although evaluating the ratio $\widetilde{p}'(\pi)/\widetilde{p}(\pi)$ is straightforward, directly using the ordinary importance-sampling correction

would still require the unknown ratio $Z_p/Z_{p'}$. Self-normalized importance sampling avoids this obstacle by estimating that ratio from the same weighted sample [18]. If $\pi^{(1)}, \dots, \pi^{(N)} \sim p$, define the importance weights

$$w^{(m)} := \frac{\widetilde{p}'(\pi^{(m)})}{\widetilde{p}(\pi^{(m)})}, \quad m \in [N].$$

Before introducing reuse, applying the direct estimator (37) under the target distribution p' would require fresh samples $\pi^{(1)}, \dots, \pi^{(N)} \sim p'$. With the importance weights above, however, the same pool $\pi^{(1)}, \dots, \pi^{(N)} \sim p$ delivers a valid self-normalized estimator of GPASV at the new target p':

$$\widehat{\psi}_i^{\mathrm{SNIS}}(U) := \frac{\sum_{m=1}^N w^{(m)} \Delta_i(\pi^{(m)}; U)}{\sum_{m=1}^N w^{(m)}}, \quad i \in [n]. \tag{41}$$

This justifies permutation reuse across nearby temperature settings. As an aside, computing only the numerator of (41) and then rescaling the resulting vector so that its coordinates sum to $U([n]) - U(\emptyset)$ recovers exactly (41) itself; this follows immediately from the efficiency axiom.

To monitor the quality of this reuse, we use the usual effective sample size [23], defined as

$$\mathrm{ESS} := \frac{\left(\sum_{m=1}^N w^{(m)}\right)^2}{\sum_{m=1}^N (w^{(m)})^2}. \tag{42}$$

For example, suppose that one first samples permutations at $(\alpha, \beta) = (0, 0)$ and then moves to a nearby target such as $(1, 0)$. Once (42) is computed and the desired Monte Carlo budget at the new target is N, one may draw only

$$N_{\mathrm{new}} := \max\{0,\, N - \lfloor \mathrm{ESS} \rfloor\}$$

fresh permutations $\widetilde{\pi}^{(1)}, \dots, \widetilde{\pi}^{(N_{\mathrm{new}})} \sim p'$. One may then combine them with the reused sample through the hybrid estimate below:

$$\widehat{\psi}_i^{\mathrm{hyb}}(U) := \frac{\mathrm{ESS}}{\mathrm{ESS} + N_{\mathrm{new}}} \frac{\sum_{m=1}^N w^{(m)} \Delta_i(\pi^{(m)}; U)}{\sum_{m=1}^N w^{(m)}} + \frac{N_{\mathrm{new}}}{\mathrm{ESS} + N_{\mathrm{new}}} \frac{1}{N_{\mathrm{new}}} \sum_{r=1}^{N_{\mathrm{new}}} \Delta_i(\widetilde{\pi}^{(r)}; U).$$

When $N_{\mathrm{new}} = 0$, the second term is omitted. This hybrid estimate interprets the reused sample as contributing roughly ESS effective draws and fills the remaining budget with fresh target samples. In this way, ESS serves as an operational measure of how much of the next budget can be inherited from the previous point in the sweep.

G Additional Details and Results for Empirical Studies in Section 5

This appendix collects the implementation details and full empirical results of the three simulation studies in Section 5. All experiments in this appendix are CPU-only and were run on the Unity high-performance computing cluster, provided by the College of Arts and Sciences at The Ohio State University across multiple Intel Xeon CPUs; the runtime-reporting experiments were pinned to Intel Xeon Platinum 8468 CPU with 128 GB RAM to keep wall-clock times comparable across configurations.

G.1 Simulation 1: Mixing Behavior of MCMC

Sampling $\pi \sim p^{(\lambda,\omega)}$ from (5) relies on an adjacent-swap Metropolis–Hastings chain (Algorithm 1), whose mixing time is not known in closed form. We provide a practical way to declare the chain mixed, describe the setup used for the grid of graphs in Section 5, explain the exact pairwise-order target against which the chain is compared, and report the full mixing-time grid that the main text only shows for one representative panel.

Throughout Section G.1 we work in the multiplicative representation of GPASV introduced in Appendix C, which absorbs the temperature β into the edge weights and makes the weak/strong-priority regimes easier to parametrize.

G.1.1 Mixing Diagnostic and Threshold Protocol

In the classical theory of Markov chains, the mixing time is defined as the first t at which the total variation between the chain's distribution at step t and its stationary distribution falls below a target threshold; once that is met, any functional of the chain is guaranteed to be close to its stationary expectation up to a uniform constant. On the permutation space Π, however, directly tracking this total variation is combinatorially out of reach: even representing the chain's current distribution over Π already requires $O(n!)$ numbers, and the stationary distribution $p^{(\lambda,\omega)}$ in (5) does not admit a closed-form normalizer. We therefore follow Talvitie et al. [39] and diagnose mixing empirically through pairwise-order probabilities and compare the resulting practical mixing times against reference growth rates from the literature.

Let $\widehat{P}_t^{(\mathrm{rand})}(i \prec j)$ and $\widehat{P}_t^{(\mathrm{greedy})}(i \prec j)$ denote the empirical probability that player i appears before player j after t proposal steps under random and greedy initialization, respectively. For each initialization scheme, we track the worst-case pairwise-order deviation from the stationary target $P^\star(i \prec j) := \mathbb{P}_{\pi\sim p^{(\lambda,\omega)}}(i \text{ appears before } j)$:

$$D_t^{(\mathrm{init})} := \max_{i,j\in[n],i\neq j} \left|\widehat{P}_t^{(\mathrm{init})}(i \prec j) - P^\star(i \prec j)\right|, \quad \mathrm{init} \in \{\mathrm{rand}, \mathrm{greedy}\}. \tag{43}$$

The quantity $D_t^{(\mathrm{init})}$ is the primary mixing diagnostic, and we report the first t at which $D_t^{(\mathrm{init})}$ falls below the target threshold $\epsilon = 1/4$ as the practical mixing time. Comparing random against greedy initialization isolates how much of the practical speedup is attributable to initialization alone.

In practice, $D_t^{(\mathrm{init})}$ is evaluated at doubling checkpoints $t = 1, 2, 4, \ldots$ up to a horizon $T(n) := \lceil n^3 \log n \rceil$, and the first crossing is localized by binary search between consecutive checkpoints. A guard band $\epsilon_0 = 0.02$ prevents Monte Carlo noise from producing spurious crossings. That is, a crossing is certified only once $D_t^{(\mathrm{init})} \leq \epsilon - \epsilon_0$, while values in $[\epsilon - \epsilon_0, \epsilon + \epsilon_0]$ are handled conservatively during the binary search. If no crossing is observed before $T(n)$, the replicate is recorded as not mixed within the simulated horizon. The greedy scheme follows Algorithm 2, and the random scheme draws the starting permutation uniformly from Π.

G.1.2 Experimental Setup

Experiments cover $n \in \{4, 6, 8, \ldots, 22\}$, which is the range where the ground-truth $P^\star$ in (43) can be computed exactly by the dynamic program described below; beyond that, exact pairwise probabilities are no longer tractable.

Graphs are drawn from two families. In the *DAG family*, each ordered pair (i, j) with $i < j$ is included as an edge independently with probability p_{edge}; the resulting graph is automatically acyclic. In the *general directed-graph family*, each ordered pair (i, j) with $i \neq j$ is included independently with probability p_{edge}, so the graph may contain cycles. We run $p_{\mathrm{edge}} \in \{0.2, 0.8, 1.0\}$ to cover sparse, dense, and saturated regimes.

Node weights are sampled as $\lambda_i \overset{\mathrm{iid}}{\sim} \mathrm{Unif}(1, U_\lambda)$ with $U_\lambda \in \{1, 100\}$; $U_\lambda = 1$ gives a homogeneous regime $\lambda_i \equiv 1$, and $U_\lambda = 100$ gives a heterogeneous regime. On present edges, the multiplicative edge weight $\widetilde{\omega}_{ij}$ is sampled as $\widetilde{\omega}_{ij} \overset{\mathrm{iid}}{\sim} \mathrm{Unif}(L_{\widetilde{\omega}}, U_{\widetilde{\omega}})$ with $(L_{\widetilde{\omega}}, U_{\widetilde{\omega}}) \in \{(0, 0.5), (0.5, 1)\}$, corresponding to a strong and a weak pairwise-priority regime respectively (recall $\widetilde{\omega}_{ij} \to 0$ is the hard limit and $\widetilde{\omega}_{ij} = 1$ is no priority). Non-edge pairs are assigned $\widetilde{\omega}_{ij} = 1$, so that they contribute a neutral factor to the stage-wise form (12).

For each parameter setting, we run 1000 independent adjacent-swap MH chains in Algorithm 1 per initialization scheme, with lazy probability $1/2$ to remove periodicity of the adjacent-swap proposal. Each setting is replicated over five graph draws and, within each graph draw, five initialization draws; the saturated graph ($p_{\mathrm{edge}} = 1$) is deterministic, so only one graph draw is used there. Within each initialization replication, the 1000 chains share the starting permutation but use independent MCMC seeds.

G.1.3 Exact Pairwise-Order Ground Truth

The diagnostic (43) requires the stationary pairwise-order probabilities $P^\star(i \prec j)$ under $p^{(\lambda,\omega)}$. Computing all $n(n-1)$ pairwise probabilities by enumeration over Π costs $O(n^2 n!)$ time, which is

Algorithm 3 Dynamic programming computation of $P^\star(i \prec j)$ under GPASV.

Require: Node weights λ, multiplicative pairwise weights $\widetilde{\omega}$
for all nonempty $S \subseteq [n]$ **do**
  for all $i \in S$ **do**
    $\widetilde{\omega}_i^S \leftarrow \prod_{r \in S \setminus \{i\}} \widetilde{\omega}_{ir}$
  end for
  for all $i \in S$ **do**
    $\ell_{\lambda,\widetilde{\omega}}(i; S) \leftarrow \lambda_i \, \widetilde{\omega}_i^S \, \big(\sum_{k \in S} \widetilde{\omega}_k^S\big) / \big(\sum_{k \in S} \lambda_k \, \widetilde{\omega}_k^S\big)$
  end for
end for
$A(\emptyset) \leftarrow 1$
for $s = 1$ to n **do**
  for all $S \subseteq [n]$ with $|S| = s$ **do**
    $A(S) \leftarrow \sum_{i \in S} A(S \setminus \{i\}) \, \ell_{\lambda,\widetilde{\omega}}(i; S)$
  end for
end for
$Z \leftarrow A([n])$ and $B([n]) \leftarrow 1$
for $s = n - 1$ down to 0 **do**
  for all $S \subseteq [n]$ with $|S| = s$ **do**
    $B(S) \leftarrow \sum_{k \notin S} \ell_{\lambda,\widetilde{\omega}}(k; S \cup \{k\}) \, B(S \cup \{k\})$
  end for
end for
Initialize $P^\star(i \prec j) \leftarrow 0$ for all i, j
for all $j \in [n]$ **do**
  for all $S \subseteq [n] \setminus \{j\}$ **do**
    $C \leftarrow A(S) \, \ell_{\lambda,\widetilde{\omega}}(j; S \cup \{j\}) \, B(S \cup \{j\}) / Z$
    for all $i \in S$ **do**
      $P^\star(i \prec j) \leftarrow P^\star(i \prec j) + C$
    end for
  end for
end for
Set the diagonal to zero and rescale complementary pairs to sum to one
return $P^\star$

already infeasible at $n = 22$. We instead adapt the subset dynamic programming (DP) of Kangas et al. [21], originally designed for the uniform distribution on linear extensions of a DAG, to the non-uniform GPASV distribution on the full Π.

Working in the multiplicative representation from Appendix C, the GPASV mass function can be written as

$$p^{(\lambda,\omega)}(\pi) = \frac{\widetilde{p}(\pi)}{Z}, \quad \widetilde{p}(\pi) = \prod_{t=1}^{n} \ell_{\lambda,\widetilde{\omega}}(\pi_t; S_t(\pi)),$$

where the stage-wise factor at step t is, reading off (11),

$$\ell_{\lambda,\widetilde{\omega}}(i; S) := \frac{\lambda_i \, \widetilde{\omega}_i^S}{\sum_{k \in S} \lambda_k \, \widetilde{\omega}_k^S} \cdot \sum_{k \in S} \widetilde{\omega}_k^S, \quad i \in S, \quad \widetilde{\omega}_k^S := \prod_{r \in S \setminus \{k\}} \widetilde{\omega}_{kr}.$$

Intuitively, $\ell_{\lambda,\widetilde{\omega}}(i; S)$ is the contribution of placing i last in prefix S: the first quotient is the conditional probability of that step, and the sum $\sum_k \widetilde{\omega}_k^S$ is the state rescaling.

Let $A(S)$ be the total unnormalized mass over all partial orders whose prefix is S, built up from shorter prefixes by inductively placing one more element last:

$$A(\emptyset) = 1, \quad A(S) = \sum_{i \in S} A(S \setminus \{i\}) \, \ell_{\lambda,\widetilde{\omega}}(i; S).$$

Then $Z = A([n])$. Similarly, let $B(S)$ be the total unnormalized mass of all completions from prefix S to the full set:

$$B([n]) = 1, \quad B(S) = \sum_{k \notin S} \ell_{\lambda,\widetilde{\omega}}(k; S \cup \{k\}) \, B(S \cup \{k\}).$$

Combining the prefix mass, the step that inserts j at the end, and the continuation mass gives, for $i \neq j$,

$$P^\star(i \prec j) = \frac{1}{Z} \sum_{\substack{S \subseteq [n] \setminus \{j\} \\ i \in S}} A(S)\, \ell_{\lambda,\widetilde{\omega}}(j; S \cup \{j\})\, B(S \cup \{j\}).$$

Each summand is the unnormalized mass of permutations whose prefix immediately before j's insertion is S; the condition $i \in S$ picks out the permutations in which i is earlier than j. After the DP, we set $P^\star(i \prec i) = 0$ and, for numerical stability at larger n, rescale each complementary pair $\{P^\star(i \prec j), P^\star(j \prec i)\}$ to sum to one.

The whole procedure runs in $O(n^2\, 2^n)$ time, replacing the $O(n^2 n!)$ enumeration cost. This is what makes exact ground-truth computation tractable up to $n = 24$. Algorithm 3 summarizes the resulting DP procedure.

G.1.4 Two Special Regimes and Reference Rates

Two limiting regimes give useful reference points for reading the mixing plot. Both correspond to the hard-priority limit $\widetilde{\omega}_{ij} \to 0$ on every present edge, which by Theorem 2 concentrates $p^{(\lambda,\omega)}$ on $\widetilde{\Pi}^{G_\omega}$.

When G_ω is a DAG and $\lambda_i \equiv 1$, $\widetilde{\Pi}^{G_\omega} = \Pi^{G_\omega}$ and the limit is uniform on the linear extensions of G_ω, which is exactly the target distribution of PSV. When G_ω is the saturated directed graph ($p_{\text{edge}} = 1$) with $\lambda_i \equiv 1$, every π incurs the same total raw violation, so the limit becomes uniform on the full Π, which is the target of the standard Shapley value. These two regimes bound the sampling difficulty within our grid and motivate the reference growth lines n^2 and $(4/\pi^2)\, n^3 \log n$ that we overlay in the mixing figures; the former is the practical mixing rate reported by Talvitie et al. [39], and the latter is the known best theoretical rate [4, 43].

G.1.5 Full Results

Figure 9 reports practical mixing times across the full grid described above. Panels separate the DAG and general directed graph families, the density p_{edge}, the node heterogeneity U_λ, and the weak/strong edge-priority regime. Greedy initialization tracks or beats random initialization throughout the grid, and the gap is largest in dense and saturated regimes, where random initialization often fails to cross the threshold within the $T(n)$ horizon. The greedy-initialized curves remain close to the n^2 reference in saturated regimes and close to the $n^3 \log n$ reference in sparse DAG regimes, matching the two limiting targets discussed above.

G.2 Simulation 2: Monte Carlo Accuracy and Estimator Comparison

Once the adjacent-swap MH chain is mixed, the remaining source of error is Monte Carlo noise in the ROV expectation (2). This subsection isolates that error by running the direct permutation estimator and the surrogate-assisted estimator from Section 4 on two synthetic families for which the exact GPASV value $\psi^\star$ is known in closed form. We also compare the three estimators, including surrogate-assisted estimators introduced in Appendix F.2.2 under matched utility-evaluation budgets.

As in Section G.1, we work in the multiplicative representation of Appendix C: the edge weight $\widetilde{\omega}_{ij} = \exp(-\beta \omega_{ij}) \in (0, 1]$, with $\widetilde{\omega}_{ij} = 1$ corresponding to no pairwise priority, $\widetilde{\omega}_{ij} \to 0$ to the hard-priority limit, and non-edge pairs set to $\widetilde{\omega}_{ij} = 1$.

G.2.1 Closed-Form Targets for Benchmark Scenarios

Both benchmarks belong to a common *sum-of-unanimity* family:

$$U(S) = \sum_{j=1}^{d} c_j\, \mathbb{1}\{T_j \subseteq S\}, \quad T_j \subseteq [n], \quad c_j \geq 0. \tag{44}$$

Under (44), the ROV marginal contribution of player $i \in T_j$ to the j-th term is c_j exactly when i is the *last* member of T_j to arrive in π, and zero otherwise. By linearity, the ROV expectation (2) therefore reduces to

$$\psi_i(U) = \sum_{j:\, i \in T_j} c_j \cdot \mathbb{P}_{\pi \sim p^{(\lambda,\omega)}} \big(i \text{ is the last member of } T_j \text{ in } \pi \big),$$

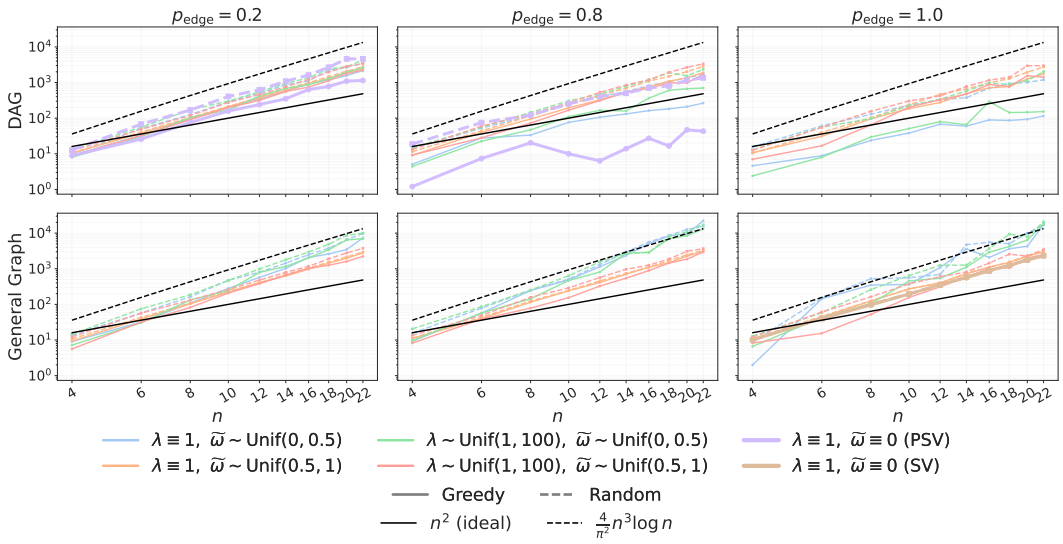


Figure 9: Practical mixing-time diagnostic for the adjacent-swap MH chain on GPASV. Solid curves use greedy initialization (Algorithm 2); dashed curves use uniform random initialization. Both axes are in log-scale, so the slope of each curve corresponds to the empirical growth rate of the mixing time. Panels vary graph family (DAG vs. general directed), density p_{edge}. Reference lines show the n^2 (practical mixing time reported by Talvitie et al. [39]) and $(4/\pi^2)n^3 \log n$ (best known theoretical mixing time reported by Bubley and Dyer [4], Wilson [43]) growth rates.

so the two scenarios differ only in how the graph and the subset family $\{T_j\}$ are chosen to make this last-arrival probability tractable.

Scenario 1: line DAG with contiguous-interval utilities. Players are totally ordered $1 \prec 2 \prec \cdots \prec n$, and every precedence edge $i < j$ shares a common multiplicative weight $\widetilde{\omega}_{ij} \equiv \widetilde{\omega}_0 \in (0, 1]$; non-edge pairs (i.e. $i > j$) take $\widetilde{\omega}_{ij} \equiv 1$. The subsets T_j in (44) are contiguous intervals $T_j = \{\ell_j, \ell_j + 1, \ldots, r_j\}$ of $[n]$. Under this structure, the last-arrival probability within each interval reduces to a λ-weighted softmax in which the total-order weight $\widetilde{\omega}_0^{r_j - i}$ discounts position $i \in T_j$.

Proposition 2 (Closed form on Scenario 1)**.** *Let the players be arranged in the total order $1 \prec 2 \prec \cdots \prec n$ with $\widetilde{\omega}_{ij} \equiv \widetilde{\omega}_0 \in (0, 1]$ for $i < j$ and $\widetilde{\omega}_{ij} \equiv 1$ on non-edge pairs. Consider a sum-of-unanimity utility (44) with $d = n^2$ and $T_j = \{\ell_j, \ell_j + 1, \ldots, r_j\}$ a contiguous interval for each j. Then the exact GPASV value is*

$$\psi_i(U) = \sum_{j:\, i \in T_j} c_j \, \frac{\lambda_i \, \widetilde{\omega}_0^{r_j - i}}{\sum_{q=\ell_j}^{r_j} \lambda_q \, \widetilde{\omega}_0^{r_j - q}}, \quad i \in [n].$$

For each interval term, player $i \in T_j$ receives c_j exactly when i is the last element of T_j to appear in π. Under the total-order priority structure, the last-arrival probability within T_j is proportional to $\lambda_i \, \widetilde{\omega}_0^{r_j - i}$; normalizing over T_j and summing over j by linearity gives the stated form. The full proof is given in Appendix E.4.

Scenario 2: block DAG with cyclic blocks and block-completion utilities. Partition $[n] = B_1 \cup \cdots \cup B_K$ into disjoint equal-sized blocks, and build a directed graph in which each block B_k carries an internal directed cycle, and whenever $B_k \to B_\ell$ is present in an underlying DAG on $\{B_1, \ldots, B_K\}$, every ordered pair $(i, j) \in B_k \times B_\ell$ is an edge. We take λ_i constant within each block; on each cycle within a block B_k we take $\widetilde{\omega}_{ij}$ to be a single common value, and on each present ordered block pair $B_k \to B_\ell$ we take $\widetilde{\omega}_{ij}$ to be a single common value over all $(i, j) \in B_k \times B_\ell$. Non-edge pairs take $\widetilde{\omega}_{ij} \equiv 1$. The subsets in (44) are the blocks themselves, $T_j = B_j$ for $j = 1, \ldots, K$, so U rewards the completion of entire blocks. Within-block cyclic symmetry, combined with equal λ's inside a block, makes every member of B_j equally likely to be the last to arrive among B_j.

Proposition 3 (Closed form on Scenario 2)**.** *Let $[n] = B_1 \cup \cdots \cup B_K$ be a partition into disjoint blocks, and let the directed graph satisfy the block-structured assumptions above: each block induces*

a directed cycle, every present block edge $B_k \to B_\ell$ contributes all pairs $(i,j) \in B_k \times B_\ell$ as edges, λ_i is constant within each block, each block's cycle edges share a common weight, each present block pair shares a common weight, and non-edge pairs take $\widetilde{\omega}_{ij} \equiv 1$. Consider the block-completion utility (44) with $d = K$ and $T_j = B_j$. Then the exact GPASV value is

$$\psi_i(U) = \frac{c_j}{|B_j|} \quad \textit{for every } i \in B_j.$$

For the block-completion term associated with B_j, only the final member of B_j to appear can receive c_j. Because node weights are constant within each block and every priority factor affecting B_j's members is symmetric under cyclic relabeling inside B_j, each member of B_j is equally likely to be this final member. The block reward thus splits uniformly within the block, and linearity over j gives the stated form. The full proof is given in Appendix E.5.

G.2.2 Experimental Design

Across both scenarios we draw iid coefficients $c_j \sim \mathrm{Unif}(0.5, 1.5)$. The graph and priority choices specialize the closed-form setups of Proposition 2 and Proposition 3 to the two-dimensional $(\lambda, \widetilde{\omega})$-regime grid used in the main text.

For **Scenario 1**, we fix $d = n^2$ and sample each interval T_j independently with replacement from the set of contiguous intervals of $[n]$. Soft priorities follow either the homogeneous regime $\lambda_i \equiv 1$ or the heterogeneous regime $\lambda_i \overset{\text{iid}}{\sim} \mathrm{Unif}(1, 10)$. The common precedence-edge weight is $\widetilde{\omega}_0 \in \{0.3,\ 0.7\}$, with $\widetilde{\omega}_0 = 0.7$ corresponding to the weak-priority regime (close to no priority) and $\widetilde{\omega}_0 = 0.3$ to the strong-priority regime.

For **Scenario 2**, we take $K = n/16$ equal-sized blocks. The between-block DAG is generated by including each ordered block pair $B_k \to B_\ell$ with $k < \ell$ independently with probability 0.8, and every block carries a fixed internal directed cycle. Soft priorities are either $\lambda_i \equiv 1$ or block-constant $\lambda_i = \lambda_{0k}$ for $i \in B_k$ with $\lambda_{0k} \overset{\text{iid}}{\sim} \mathrm{Unif}(1, 10)$. Edge priorities are block-structured: for each present block pair $B_k \to B_\ell$, the pair-level common weight $\widetilde{\omega}_{ij} \equiv \widetilde{\omega}^{\text{bet}}_{k\ell}$ over all $(i,j) \in B_k \times B_\ell$, and within each block B_k the cycle edges share a common weight $\widetilde{\omega}^{\text{cyc}}_k$. Both $\widetilde{\omega}^{\text{bet}}_{k\ell}$ and $\widetilde{\omega}^{\text{cyc}}_k$ are drawn either from $\mathrm{Unif}(0.5, 1)$ (weak-priority regime) or from $\mathrm{Unif}(0, 0.5)$ (strong-priority regime).

Case 1–4 regimes. The two λ-regimes and the two $\widetilde{\omega}$-regimes factor into four cases, shared between the two scenarios so that the ARE, AUCC, unique-subset, and runtime reports use the same column layout. The homogeneous λ-regime is identical across scenarios, while the heterogeneous one differs: Scenario 1 uses a per-player draw, whereas Scenario 2 uses the block-constant version $\lambda_i = \lambda_{0k}$ for $i \in B_k$.

Case	Scenario 1		Scenario 2	
	λ_i	$\widetilde{\omega}_0$ $(i < j)$	λ_i (for $i \in B_k$)	$\widetilde{\omega}^{\text{bet}}_{k\ell}$, $\widetilde{\omega}^{\text{cyc}}_k$
1	$\equiv 1$	$\equiv 0.7$	$\equiv 1$	$\sim \mathrm{Unif}(0.5, 1)$
2	$\equiv 1$	$\equiv 0.3$	$\equiv 1$	$\sim \mathrm{Unif}(0, 0.5)$
3	$\sim \mathrm{Unif}(1, 10)$	$\equiv 0.7$	$\equiv \lambda_{0k} \sim \mathrm{Unif}(1, 10)$	$\sim \mathrm{Unif}(0.5, 1)$
4	$\sim \mathrm{Unif}(1, 10)$	$\equiv 0.3$	$\equiv \lambda_{0k} \sim \mathrm{Unif}(1, 10)$	$\sim \mathrm{Unif}(0, 0.5)$

G.2.3 Accuracy Metrics and MH Protocol

We report results for $n \in \{32, 128, 512, 2048\}$ with a total Monte Carlo budget of $N_{\text{MC}} = 20{,}000$ post-burn-in permutations. For each (n, Case), we fix one synthetic instance and run 10 independent MH chains with greedy initialization (Algorithm 2), burn-in $\lceil n^{2.5} \rceil$, and thinning interval 1000. Standard deviations reported throughout Section G.2 are taken across these 10 repetitions on the fixed instance.

Every 100 samples we record the absolute relative error

$$\mathrm{ARE}(m) := \frac{\|\widehat{\psi}^{(m)} - \psi^\star\|_2}{\|\psi^\star\|_2},$$

where $\widehat{\psi}^{(m)}$ is the direct Monte Carlo estimate of ψ after m post-burn-in samples and $\psi^\star$ is the closed-form target from Proposition 2 or 3. We summarize the whole convergence path by the area under the convergence curve,

$$\mathrm{AUCC} := \frac{1}{200}\sum_{\ell=1}^{200}\mathrm{ARE}(100\ell),$$

which penalizes slow initial convergence and high terminal error on a common scale. Every 100 samples we also record the number of distinct subsets whose utility $U(S)$ has been evaluated up to that point, which measures cache reuse, and the cumulative non-utility runtime, which isolates sampling and bookkeeping cost from the utility evaluation.

Concretely, we fix $\lambda_i \equiv 1$ and sweep two axes: the common edge weight $\widetilde{\omega}_0 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and the burn-in exponent c in n^c with $c \in \{0, 0.5, 1, 1.5, 2, 2.5\}$. All other MH settings match the main Section G.2 protocol. For each $(\widetilde{\omega}_0, c)$ we run both random and greedy initialization and report AUCC. This ablation separates two questions: how much of the greedy benefit is simply a shorter burn-in substitute, and how that tradeoff depends on the pairwise-priority strength.

G.2.4 Matched-Budget Surrogate Comparison Protocol

Beyond the direct permutation estimator, GPASV admits a surrogate-assisted variant that first fits a cheap surrogate $\widehat{h} \approx U$ on a limited set of evaluated subsets and then corrects the surrogate's GPASV by a residual term; see Section F.2.2 for the full construction. We compare three estimators:

(i) the direct permutation estimator of Section 4;

(ii) a linear surrogate $\widehat{h}(S) = U(\emptyset) + \sum_{i\in S}\widehat{a}_i$, in which each player contributes additively;

(iii) a quadratic surrogate $\widehat{h}(S) = U(\emptyset) + \sum_{i\in S}\widehat{a}_i + \sum_{i,j\in S}\widehat{b}_{ij}$ that includes a randomly chosen 10% of the pairwise interactions.

Since the dominant cost of GPASV is direct utility evaluation, we compare the estimators by matching them on the number of distinct evaluated subsets. For each problem instance we first run the direct estimator and record the number K_{eval} of distinct subsets whose $U(S)$ was evaluated. The surrogate estimators are then given the same budget: $K_{\mathrm{train}} := \min\{\lfloor 0.2\, K_{\mathrm{eval}}\rfloor, 200{,}000\}$ evaluations go toward fitting $\widehat{h}$, and the remaining $K_{\mathrm{adjust}} := K_{\mathrm{eval}} - K_{\mathrm{train}}$ evaluations go toward the residual correction. The two phases share the same utility cache, so K_{eval} is counted over the union of evaluated subsets rather than the sum. Permutation samples that do not require utility evaluation, namely those used to estimate subset coefficients and pairwise-order probabilities, do not count toward K_{eval}; we use $M = 100{,}000$ such utility-free permutation samples.

Problem sizes are $n \in \{64, 128, 256, 512, 1024\}$, and all other MH settings (burn-in, thinning, greedy initialization) match the main Section G.2 setup.

G.2.5 Full Results

Direct permutation estimator. Figure 10 reports $\mathrm{ARE}(m)$ over the sampling budget on the shared 2×4 Scenario$\times$Case grid; Table 3 reports the final AUCC; Figure 11 reports the number of distinct coalition subsets evaluated; Table 4 reports the final non-utility runtime. Error drops with the sampling budget throughout the grid, and the slope flattens as n grows. Scenario 2 is harder than Scenario 1 at matched n, consistent with the extra mixing burden from within-block directed cycles. The unique-subset curves flatten as m grows, reflecting increasing cache reuse along each chain.

Surrogate-assisted estimators. Figure 12 compares the direct permutation estimator with the two surrogate variants under matched utility-evaluation budgets. Surrogate methods reduce ARE in most settings, and the quadratic surrogate tends to dominate on Scenario 2 where pairwise interactions are informative. Table 5 and Table 6 report the non-utility runtime and the surrogate training memory. The gain in ARE is bought at a substantial computational cost: quadratic-surrogate runtime and memory both grow quickly with n. Unlike PASV, which can zero out b_{ij} for any pair that violates the DAG, the GPASV quadratic surrogate has to retain all sampled pairwise coefficients, so memory is harder to trim.

G.3 Simulation 3: Priority Sweeping

Simulations 1 and 2 study the computational side of GPASV. Simulation 3 asks a different question: how does the attribution respond when we move a single soft-priority scalar along a sweep, and how does that response differ between GPASV and PASV? This is the empirical counterpart to the priority-sweeping diagnostic introduced in Section 3.5, and it illustrates an expected consequence of GPASV's full support: ω at finite β leaves more room for λ to reshape the order distribution than PASV's binary DAG does.

G.3.1 Setup: Group Selection and Utility Functions

We fix $n = 32$ and for each $p_{\text{edge}} \in \{0.2, 0.5, 0.8\}$, draw a single DAG on $[n]$ by including each precedence edge $i < j$ independently with probability p_{edge}. We then fix a single group $H \subseteq [n]$ with $|H| = n/2$, and sweep two parameters: the soft-priority level λ_0 on H and the hard-priority strength β. Throughout, we write π_i^{-1} for the position of player i in the permutation π, i.e. the unique $t \in [n]$ with $\pi_t = i$.

Soft-priority sweep. We set $\lambda_i = \lambda_0$ for $i \in H$ and $\lambda_i = 1$ otherwise, and sweep

$$\lambda_0 \in \{1, 2, 4, 8, 16, 32, 64, 128\}.$$

At $\lambda_0 = 1$ all players share the same soft priority, so GPASV and PASV both assign the same value to H as to $[n] \setminus H$ up to graph effects; larger λ_0 progressively up-weights the members of H in the backward stage-wise softmax, making them more likely to be selected for later forward positions, hence pushing them toward the tail of π.

Hard-priority sweep. We vary the temperature β in (5) over

$$\beta \in \{1, 2, 4, 8\},$$

and include the hard-priority extreme $\beta \to \infty$ (which, on a DAG, reduces GPASV to PASV by Theorem 2) as a reference curve.

Utility functions. We again use the sum-of-unanimity (SOU) family from Section G.2, but now with T_k ranging over general nonempty subsets of $[n]$ rather than contiguous intervals. Concretely, we draw $d = n^2$ subsets $T_1, \ldots, T_d$ iid uniformly from the nonempty subsets of $[n]$ together with iid coefficients $c_k \sim \text{Unif}(0.5, 1.5)$, and set

$$U^{\text{SOU}}(S) := \sum_{k=1}^{d} c_k \, \mathbb{1}\{T_k \subseteq S\}.$$

For each k, the marginal value of player $i \in T_k$ under the indicator $\mathbb{1}\{T_k \subseteq S\}$ is

$$\mathbb{P}_{\pi \sim p^{(\lambda,\omega)}}\big(i \text{ is the last member of } T_k \text{ in } \pi\big).$$

Hence U^{SOU} rewards late-positioned players.

To probe the opposite end of the position axis, we additionally introduce the *sum-of-race* (SOR) family, obtained by replacing the indicator of $T_k \subseteq S$ with that of $T_k \cap S \neq \emptyset$:

$$U^{\text{SOR}}(S) := \sum_{k=1}^{d} c_k \, \mathbb{1}\{T_k \cap S \neq \emptyset\}.$$

For each k, the marginal value of player $i \in T_k$ under $\mathbb{1}\{T_k \cap S \neq \emptyset\}$ is

$$\mathbb{P}_{\pi \sim p^{(\lambda,\omega)}}\big(i \text{ is the first member of } T_k \text{ in } \pi\big),$$

so U^{SOR} rewards early-positioned players. We write U^{SOU} and U^{SOR} throughout to distinguish the two utilities. Running the same sweep on U^{SOU} and U^{SOR} therefore exposes λ_0-sensitivity from two opposite ends.

G.3.2 Connection to Limiting Cases

Two extremes of β bracket the sweep curves and help interpret the intermediate β's.

$\beta \to \infty$ **(hard-priority limit).** By Theorem 2, the GPASV distribution concentrates on $\widetilde{\Pi}^{G_\omega}$, and when G_ω is a DAG we have $\widetilde{\Pi}^{G_\omega} = \Pi^{G_\omega}$ together with $M_\omega(S) = \max(S)$. The stage-wise factors in (5) then reduce to PASV's stage-wise factors in (3), so the group-sum curve at $\beta \to \infty$ coincides with the PASV group-sum on the same DAG. This is the dashed reference shown in the sweep plots. Under PASV, the impact of λ_0 is bounded by the hard constraints of G_ω: once the DAG order is satisfied, λ can only redistribute within admissible sub-permutations, which already fixes most of the relative order of H against $[n] \setminus H$.

$\beta = 0$ **(no pairwise priority).** When $\beta = 0$, the graph term V_ω drops out of (5) and the distribution reduces to the Plackett–Luce distribution with worths (λ_i) (Section 3.3). In this regime, λ_0 has the largest room to move the group-sum attribution because there is no hard constraint to pull the order back; the position of each member of H is determined by the worths alone.

Intermediate $\beta \in \{1, 2, 4, 8\}$ interpolates between the two extremes. Because GPASV softens the hard priority at finite β, λ_0 can push members of H later even when such moves violate an edge of G_ω; the softness is controlled by β.

G.3.3 Implementation and Baseline Computation

For each (λ_0, β) pair with finite β, we sample $N_{\mathrm{MC}} = 10{,}000$ after $10,000$ burn-in periods from $p^{(\lambda,\omega)}$ using the adjacent-swap MH sampler (Algorithm 1) with greedy initialization (Algorithm 2). We replicate each setting over 10 independent chains and report mean and standard deviation. Along the λ_0 sweep at fixed β, we reuse previous samples through the SNIS rule described in Section F.3.2, monitoring the effective sample size and drawing additional fresh samples when it falls below target. This keeps the number of utility evaluations manageable across the sweep.

The PASV reference is obtained by sampling PASV permutations directly on Π^{G_ω} following the adjacent-swap MH sampler of Lee et al. [26] and estimating the group-sum PASV by the same Monte Carlo averaging, with matched sample count.

For each sample we record both the position π_i^{-1} of every member $i \in H$, used to compute the mean forward position

$$\bar{r}_H(\lambda_0, \beta) := \frac{1}{|H|} \sum_{i \in H} \mathbb{E}_{\pi \sim p^{(\lambda,\omega)}}[\pi_i^{-1}],$$

by its empirical estimate and the marginal contributions under U^{SOU} and U^{SOR}, used to form the group-sum GPASV $\sum_{i \in H} \psi_i(U)$.

G.3.4 Full Results

Figure 13 shows the full sweep. The left panel reports $\bar{r}_H(\lambda_0, \beta)$, the middle and right panels report $\sum_{i \in H} \psi_i(U)$ under U^{SOU} and U^{SOR}, and the dashed black curve in each panel is the PASV reference.

Three patterns are visible. First, the mean forward position is monotone in λ_0: larger λ_0 pushes members of H later in the sampled order under both GPASV and PASV. Second, the slope of the position curve is steeper at smaller β, reflecting softer pairwise priority; at $\beta = 1$, λ_0 moves the mean position over a wider range than at $\beta = 8$ or at $\beta \to \infty$ (PASV). Third, the group-sum curves under U^{SOU} and U^{SOR} move in opposite directions, as expected from their definitions: the U^{SOU} group sum rises with λ_0 (more H-members arrive late and thus collect unanimity rewards), whereas the U^{SOR} group sum falls with λ_0 (fewer H-members arrive early and thus collect first-arrival rewards). The transition sharpens monotonically in β, and the $\beta \to \infty$ reference tracks the $\beta = 8$ curve closely throughout, consistent with Theorem 2.

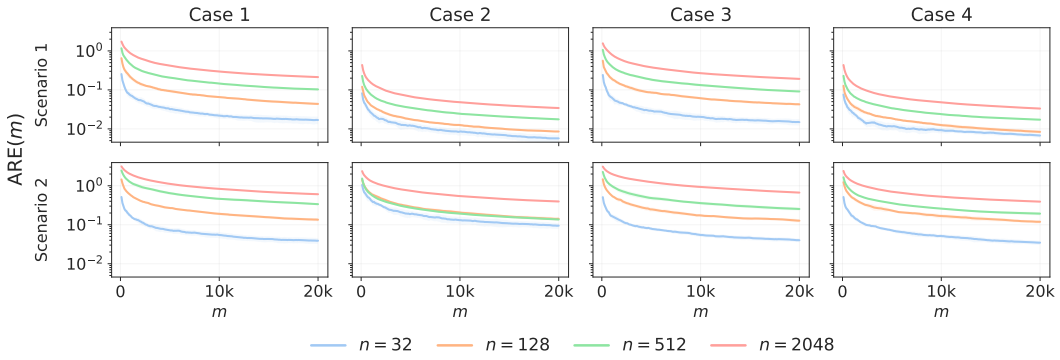


Figure 10: Monte Carlo accuracy of the direct permutation estimator, reported as $\mathrm{ARE}(m)$ over the post-burn-in sampling budget m. All curves use greedy initialization. Rows correspond to Scenario 1 and Scenario 2; columns correspond to Cases 1–4.

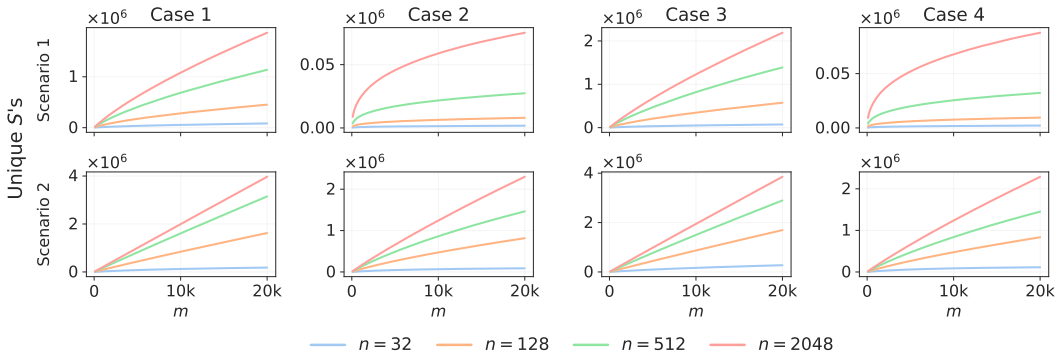


Figure 11: Number of distinct coalition subsets evaluated by the direct permutation estimator, over the sampling budget. Flatter growth indicates stronger cache reuse.

Table 3: Final AUCC for the direct permutation estimator under greedy initialization. Rows index n; columns index Cases 1–4 within each scenario. Entries are mean (standard deviation) across 10 independent MH repetitions on the fixed instance.

	Scenario 1				**Scenario 2**			
n	Case 1	Case 2	Case 3	Case 4	Case 1	Case 2	Case 3	Case 4
32	0.0310 (0.0024)	0.0111 (0.0012)	0.0278 (0.0033)	0.0115 (0.0011)	0.0708 (0.0075)	0.1772 (0.0249)	0.0741 (0.0050)	0.0681 (0.0048)
128	0.0875 (0.0052)	0.0167 (0.0006)	0.0811 (0.0040)	0.0170 (0.0010)	0.2491 (0.0076)	0.2685 (0.0116)	0.2361 (0.0144)	0.2160 (0.0151)
512	0.1919 (0.0072)	0.0326 (0.0006)	0.1726 (0.0085)	0.0319 (0.0008)	0.5831 (0.0310)	0.2540 (0.0078)	0.4506 (0.0327)	0.3246 (0.0079)
2048	0.3799 (0.0129)	0.0641 (0.0012)	0.3465 (0.0099)	0.0634 (0.0009)	0.9900 (0.0178)	0.6552 (0.0113)	1.0791 (0.0274)	0.6352 (0.0076)

Table 4: Final non-utility runtime (seconds) for the direct permutation estimator under greedy initialization. This excludes the time spent evaluating $U(S)$ and isolates sampling plus bookkeeping cost. Entries are mean (standard deviation) across 10 independent MH repetitions.

	Scenario 1				**Scenario 2**			
n	Case 1	Case 2	Case 3	Case 4	Case 1	Case 2	Case 3	Case 4
32	4.37 (0.50)	3.15 (0.07)	4.26 (0.01)	3.31 (0.01)	2.82 (0.01)	2.54 (0.01)	3.14 (0.01)	2.71 (0.01)
128	10.41 (0.08)	8.48 (0.01)	10.88 (0.05)	8.72 (0.01)	10.89 (0.09)	8.57 (0.02)	10.61 (0.02)	8.70 (0.18)
512	45.52 (0.47)	38.23 (0.26)	43.10 (0.23)	39.08 (0.20)	47.98 (0.62)	38.91 (0.34)	47.34 (0.67)	38.86 (0.34)
2048	1240.88 (49.17)	1199.02 (64.10)	1195.87 (57.89)	1232.67 (66.66)	1503.55 (53.94)	1045.47 (8.85)	1205.56 (24.25)	993.38 (25.95)

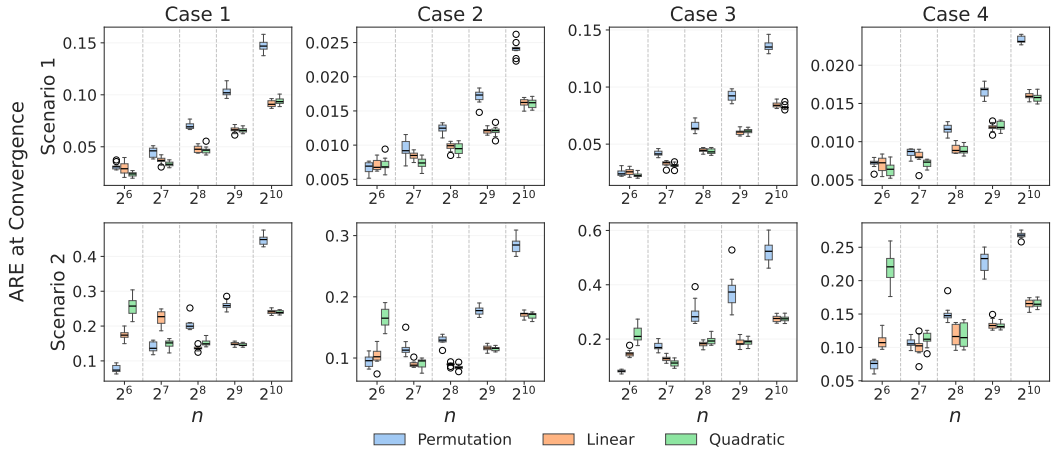


Figure 12: Final ARE under matched utility-evaluation budgets for the direct and the two surrogate-assisted estimators.

Table 5: Non-utility runtime (seconds) for the three estimators under matched utility budgets. Entries are mean (standard deviation) across independent repetitions.

Method	n	Scenario 1 Case 1	Scenario 1 Case 2	Scenario 1 Case 3	Scenario 1 Case 4	Scenario 2 Case 1	Scenario 2 Case 2	Scenario 2 Case 3	Scenario 2 Case 4
Permutation									
	64	5.98 (0.02)	4.94 (0.01)	6.31 (0.01)	5.17 (0.01)	5.21 (0.01)	4.69 (0.02)	5.28 (0.02)	4.68 (0.02)
	128	9.73 (0.01)	8.53 (0.02)	10.13 (0.01)	8.80 (0.01)	10.13 (0.02)	8.15 (0.04)	10.89 (0.10)	8.26 (0.02)
	256	17.72 (0.03)	15.94 (0.03)	18.12 (0.05)	16.21 (0.02)	19.37 (0.06)	15.99 (0.07)	19.65 (0.16)	15.94 (0.04)
	512	39.77 (0.19)	36.45 (0.16)	39.68 (0.10)	36.67 (0.12)	41.84 (0.15)	34.36 (0.06)	42.44 (0.17)	34.03 (0.15)
	1024	209.25 (3.47)	195.76 (2.46)	203.22 (1.79)	196.92 (1.99)	254.92 (3.92)	205.55 (0.51)	255.72 (3.14)	206.43 (1.47)
Linear surrogate									
	64	36.40 (0.26)	26.51 (0.15)	38.81 (0.23)	27.21 (0.19)	44.20 (0.39)	35.50 (0.38)	44.38 (0.24)	35.58 (0.26)
	128	66.61 (0.34)	46.72 (0.19)	69.92 (0.19)	48.57 (0.23)	91.95 (0.24)	61.27 (0.28)	98.72 (0.20)	61.63 (0.36)
	256	131.41 (0.51)	91.73 (0.17)	133.43 (0.76)	92.27 (0.53)	174.25 (1.43)	120.17 (0.39)	180.50 (0.31)	117.79 (0.51)
	512	290.04 (1.83)	191.07 (0.65)	272.53 (1.29)	188.95 (0.92)	309.91 (2.41)	231.47 (1.18)	311.12 (1.90)	226.24 (1.08)
	1024	1053.69 (19.83)	663.81 (6.91)	835.67 (4.70)	649.31 (4.14)	886.68 (10.01)	725.34 (4.64)	893.16 (6.54)	735.43 (3.17)
Quadratic surrogate									
	64	36.77 (0.25)	26.49 (0.10)	39.15 (0.25)	27.15 (0.21)	45.32 (0.45)	36.08 (0.24)	45.28 (0.34)	36.26 (0.25)
	128	68.86 (0.19)	46.89 (0.26)	71.97 (0.24)	48.63 (0.16)	95.70 (0.36)	63.95 (0.22)	103.02 (0.61)	65.00 (0.33)
	256	144.39 (0.74)	92.30 (0.28)	152.45 (1.20)	92.73 (0.18)	191.26 (1.16)	138.65 (0.88)	198.05 (0.71)	138.68 (1.59)
	512	359.25 (3.06)	194.82 (0.74)	342.59 (2.79)	193.55 (0.82)	382.64 (1.43)	317.75 (3.33)	388.41 (2.06)	315.09 (2.61)
	1024	1599.66 (23.32)	700.01 (6.31)	1558.86 (18.52)	701.05 (4.81)	1520.96 (14.39)	1378.02 (9.58)	1575.88 (16.45)	1409.52 (10.56)

Table 6: Training memory (GB) used by the surrogate fit under matched utility budgets. The permutation estimator does not fit a surrogate and is omitted.

Method	n	Scenario 1 Case 1	Scenario 1 Case 2	Scenario 1 Case 3	Scenario 1 Case 4	Scenario 2 Case 1	Scenario 2 Case 2	Scenario 2 Case 3	Scenario 2 Case 4
Linear surrogate									
	64	0.033 (0.000)	0.001 (0.000)	0.037 (0.000)	0.001 (0.000)	0.138 (0.000)	0.093 (0.000)	0.139 (0.000)	0.092 (0.000)
	128	0.132 (0.001)	0.002 (0.000)	0.155 (0.002)	0.003 (0.000)	0.298 (0.000)	0.194 (0.000)	0.289 (0.000)	0.183 (0.000)
	256	0.443 (0.003)	0.009 (0.000)	0.547 (0.004)	0.011 (0.000)	0.580 (0.001)	0.592 (0.000)	0.580 (0.001)	0.568 (0.006)
	512	1.151 (0.001)	0.031 (0.000)	1.144 (0.002)	0.036 (0.000)	1.146 (0.001)	1.158 (0.001)	1.144 (0.002)	1.155 (0.001)
	1024	2.294 (0.003)	0.106 (0.001)	2.298 (0.004)	0.126 (0.001)	2.304 (0.003)	2.312 (0.002)	2.297 (0.003)	2.318 (0.002)
Quadratic surrogate									
	64	0.094 (0.001)	0.002 (0.000)	0.102 (0.001)	0.002 (0.000)	0.409 (0.001)	0.290 (0.001)	0.409 (0.000)	0.288 (0.000)
	128	0.666 (0.008)	0.012 (0.000)	0.772 (0.009)	0.015 (0.000)	1.526 (0.003)	0.986 (0.003)	1.474 (0.003)	0.915 (0.002)
	256	4.030 (0.024)	0.081 (0.001)	4.922 (0.041)	0.098 (0.001)	5.305 (0.008)	5.620 (0.008)	5.336 (0.011)	5.394 (0.061)
	512	20.592 (0.042)	0.566 (0.006)	20.403 (0.041)	0.657 (0.006)	20.549 (0.034)	21.000 (0.027)	20.477 (0.041)	20.989 (0.042)
	1024	106.818 (0.211)	3.724 (0.031)	107.120 (0.217)	4.409 (0.035)	107.556 (0.187)	108.276 (0.155)	107.107 (0.175)	108.548 (0.215)

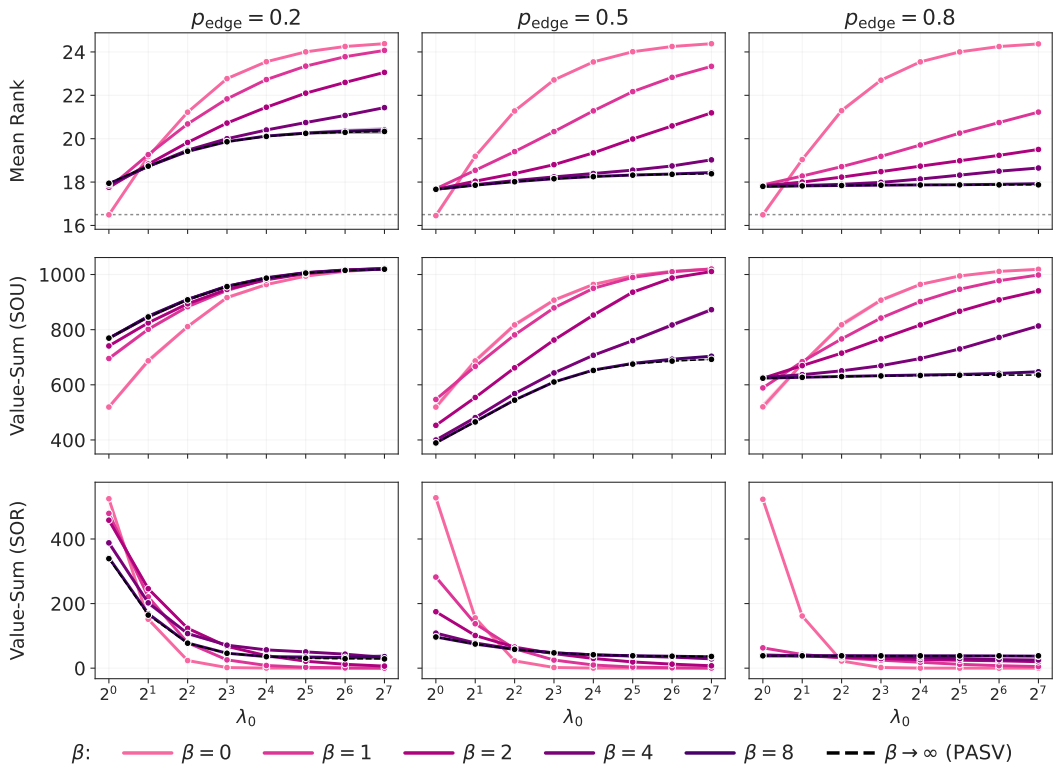


Figure 13: Priority sweeping results. Rows: mean forward position, SOU group-sum, and SOR group-sum. Colored curves vary $\beta \in \{1, 2, 4, 8\}$; the dashed black curve is the $\beta \to \infty$ (PASV on G_ω) reference.

H Additional Details and Results for LLM Evaluation in Section 6

This appendix collects the implementation details and the full empirical results of the LLM evaluation in Section 6. Throughout, a *player* i is one of the $n = 20$ LLMs in Table 7 and a *coalition* $S \subseteq [n]$ is a subset of them; GPASV attributes the coalition utility $U_{\text{ens}}(S)$ across players under the hard-priority weights ω_{ij} and the soft-priority vector $\lambda_i = \exp(-\alpha z_i)$ from Section 6.

The utility-evaluation stage of this experiment requires GPU computing and was run on the same computing cluster as Appendix G, pinned to NVIDIA H200 GPUs serving `Qwen3.5-35B-A3B-FP8` via vLLM, for a total of approximately 30 GPU-days across the 80 MT-Bench prompts. All downstream stages are CPU-only, and were run on the same cluster on multiple Intel Xeon CPUs.

H.1 Data and Priority Construction

Model set and open/paid labels. Table 7 lists the 20 models used as players in Section 6, together with the binary label $z_i \in \{0, 1\}$ used for the soft priority. The model set is exactly the intersection of MT-Bench [46] and Chatbot Arena [7]: every model we score on MT-Bench must also have a sufficient number of Chatbot Arena comparisons for the hard priority to be well-defined. We label the five proprietary API models (`gpt-4`, `gpt-3.5-turbo`, `claude-v1`, `claude-instant-v1`, `palm-2`) as *paid*, and all remaining MT-Bench/Arena-overlap models as *open-source*. The label is a coarse deployment signal rather than a license audit; the point of the soft priority in Section 6 is merely to study how strongly an evaluator who explicitly prefers open-source models over paid APIs reshapes the attribution, so z_i is set to 1 for open-source and 0 for paid.

Model	Label	Model	Label
`gpt-4`	paid	`gpt4all-13b-snoozy`	open-source
`claude-v1`	paid	`mpt-7b-chat`	open-source
`claude-instant-v1`	paid	`rwkv-4-raven-14b`	open-source
`gpt-3.5-turbo`	paid	`alpaca-13b`	open-source
`guanaco-33b`	open-source	`oasst-pythia-12b`	open-source
`vicuna-13b`	open-source	`fastchat-t5-3b`	open-source
`wizardlm-13b`	open-source	`chatglm-6b`	open-source
`palm-2`	paid	`stablelm-tuned-alpha-7b`	open-source
`vicuna-7b`	open-source	`dolly-v2-12b`	open-source
`koala-13b`	open-source	`llama-13b`	open-source

Table 7: Model set and open/paid labels z_i used in the LLM evaluation experiment.

Coalition utility from MT-Bench. MT-Bench consists of $Q = 80$ two-turn prompts organized into eight categories with ten prompts each: `Writing`, `Roleplay`, `Reasoning`, `Math`, `Coding`, `Extraction`, `Knowledge I`, and `Knowledge II`. The released benchmark bundles first- and second-turn model responses for every player, and these responses are treated as fixed candidate answers throughout the experiment; we never re-query the 20 players themselves. Following the MT-Bench judging protocol, we use two judge variants. For `Reasoning`, `Math`, and `Coding`, the judge is given MT-Bench's released reference answer and is asked to score correctness relative to it (with-reference variant). For the remaining five categories the judge scores the answer directly from the user prompt and the assistant response (no-reference variant). For a coalition S and a prompt q, the first-turn candidates $\{y_i^{(1)}\}_{i \in S}$ are passed to an aggregator LLM that synthesizes a single ensemble answer $y_S^{(1)}$; a judge LLM scores $y_S^{(1)}$ on $\{1, \ldots, 10\}$, producing a score $r^{(1)}(S, q)$. For the second turn we feed the first-turn aggregated answer $y_S^{(1)}$ as the previous assistant turn, aggregate the second-turn candidates into $y_S^{(2)}$, and score analogously, producing $r^{(2)}(S, q)$. The prompt-level score is the mean of the two turn scores, and the coalition utility is the average over prompts,

$$U_{\text{ens}}(S) = \frac{1}{Q} \sum_{q=1}^{Q} \frac{r^{(1)}(S, q) + r^{(2)}(S, q)}{2}, \qquad U_{\text{ens}}(\emptyset) := 0.$$

Hard-priority graph from Chatbot Arena. Chatbot Arena supplies pairwise human preferences between model responses; we use the public comparison counts to construct the hard priority. For

every ordered pair (i, j) with at least 50 recorded comparisons, let $\widehat{p}_{ij}$ denote the empirical win probability of i over j, and let i denote the majority-preferred side so that $\widehat{p}_{ij} \geq 1/2$. We then set

$$\omega_{ij} = \widehat{p}_{ij} - \tfrac{1}{2}, \qquad \omega_{ji} = 0,$$

so that the additive priority on edge $i \to j$ encodes how strongly the Arena crowd prefers i over j. Pairs with fewer than 50 comparisons are treated as non-edges ($\omega_{ij} = \omega_{ji} = 0$). Larger β in (5) more strongly suppresses permutations that place a majority-losing model before its majority-preferred counterpart. The resulting graph G_ω is *not* a DAG: majority preferences form several intransitive triangles among mid-tier models, which is precisely the regime that motivates GPASV over PASV.

Input scales. Following the pre-processing recommendation in Section 3.2, Table 8 reports the numerical scales of the two priority inputs. Both lie in $[0, 1]$, and we use them without further rescaling.

	SD	Min	Max
Raw soft priority score z_i	0.433	0	1
Edge weight ω_{ij}	0.110	0	0.446

Table 8: Numerical scales of the priority inputs in the LLM evaluation experiment.

H.2 Aggregator–Judge Pipeline

Model choice and inference setup. Both the aggregator and the judge are instantiated with `Qwen3.5-35B-A3B-FP8` [37]. The model is chosen because (i) it is strong enough to aggregate and score MT-Bench responses reliably yet (ii) small enough in FP8 form to be served locally with batched `vLLM` inference [24], which is essential at the cached-computation scale described in Section H.3. Using a single model for both roles removes judge–aggregator style mismatches. We cap generation at 4096 new tokens per call, and sample with temperature 1.0, top-$p = 0.95$, top-$k = 20$, following the official Qwen3.5 recommendation for instruct (non-thinking) mode on reasoning tasks. Every realized (S, q) output is cached and reused across all priority regimes, so all priority comparisons are made against the same fixed utility realization.

Prompt templates. The aggregator and judge prompt templates are shown in Appendix H.6; we summarize their structure here. The aggregator receives the user prompt together with a randomly shuffled list of candidate responses, each delimited by explicit start/end markers, and is instructed to merge, edit, and reorganize the candidates while preserving the user's requested format and without adding external facts. For the second turn, the first-turn aggregated answer $y_S^{(1)}$ is inserted as the previous assistant message before presenting the second-turn candidates. The no-reference judge prompt consists of an `[Instruction]` block, the `[Question]` block, and the assistant answer delimited by start/end markers; the with-reference variant additionally inserts a reference-answer block and asks the judge to evaluate correctness against it. For second-turn judging, the full two-turn conversation is presented in the `Assistant A` format, so that coherence across turns is scored rather than each turn in isolation. In all variants, the judge is asked for a brief explanation followed by a final `[[rating]]` on the 1–10 scale. Placeholders `{x1}`, `{x2}`, `{y1S}`, `{y2S}`, `{r1}`, `{r2}`, and `{candidate answer` k`}` are filled by the corresponding MT-Bench prompt, aggregated response, reference answer, or candidate-model response at runtime.

H.3 Sampling and Computation Protocol

MH hyperparameters. For every priority regime (α, β) with $\beta > 0$, we draw $N_{\mathrm{MC}} = 1000$ permutations from $p^{(\lambda,\omega)}$ using the adjacent-swap Metropolis–Hastings sampler of Section F.1.1 with greedy initialization (Algorithm 2). We use burn-in 10^5 and thinning interval 10^3; at $n = 20$ the adjacent-swap chain mixes quickly relative to the simulation-study scales, so this conservative schedule leaves the effective sample size close to N_{MC}. At the baseline $(\alpha, \beta) = (0, 0)$ the sampler reduces to uniform sampling over Π, which we implement by direct iid uniform draws.

Utility cache design. Evaluating one coalition on the full MT-Bench set requires four LLM calls per prompt: aggregation and judging for each of the two turns. A naive direct Monte Carlo over

N_{MC} permutations of n players with Q prompts would therefore require $4N_{\mathrm{MC}}nQ$ calls, which at $N_{\mathrm{MC}} = 1000$, $n = 20$, $Q = 80$ amounts to 6.4×10^6 calls for a *single* priority regime, and across the 19 regimes of Section 6 it would scale linearly in the number of regimes. We avoid this by caching at the level of (subset, prompt) pairs: each cache row is keyed by a 20-bit subset mask and an MT-Bench prompt index, and stores both aggregated answers $y_S^{(1)}, y_S^{(2)}$, both raw judge responses, and the two parsed ratings. Because GPASV only evaluates U_{ens} on prefix coalitions of sampled permutations, and because neighboring priority regimes along each sweep share most prefixes through the SNIS permutation reuse described next, the realized number of distinct subset evaluations is far below the worst-case $N_{\mathrm{MC}}n$. Once the cache has been populated along the first sweep, all subsequent priority regimes read the same cache: changing (α, β) changes only the sampled order distribution, not U_{ens}.

Sweep traversal and SNIS/ESS reuse. The three sweeps of Section 6 are traversed in increasing temperature order from the shared baseline:

$$(0,0) \to (1,0) \to (2,0) \to (4,0) \to (8,0) \to (16,0) \to (32,0),$$

and analogously for the β-only and joint $\alpha = \beta$ sweeps. Between neighboring regimes we apply the SNIS/ESS reuse rule from Section F.3.2. Given the reweighted effective sample size ESS of the previous-regime samples under the new target, we add

$$N_{\mathrm{new}} = \min\{N_{\mathrm{MC}},\ \max(500,\ \lceil N_{\mathrm{MC}} - \mathrm{ESS} \rceil)\}$$

fresh permutations from the new target and combine the reweighted estimate with the fresh Monte Carlo estimate using weights proportional to ESS and N_{new}. The lower bound of 500 ensures that even near-overlapping regimes refresh a meaningful fraction of the sample; the upper bound of N_{MC} caps the cost of any single transition at a full fresh draw.

Realized acceleration. Table 9 reports the realized utility-evaluation cost under the four combinations of the two reuse mechanisms: subset reuse via the cache and permutation reuse via SNIS. The cache provides the dominant fraction of the saving, reducing the per-prompt distinct subset count by a factor of ≈ 4.9 on its own (B vs. A); SNIS trims the effective permutation count from 19,000 to 15,840 along the three sweeps (Table 10) but on its own only yields a $\approx 1.2\times$ reduction (C vs. A). Combining both gives a realized $\approx 5.4\times$ reduction over the naive baseline (D vs. A).

Wall-clock cost and scaling. One un-cached evaluation of $U_{\mathrm{ens}}(S)$ requires 320 LLM calls and takes 32.21 seconds on average. We used two NVIDIA H200 GPUs in parallel; almost all runtime is spent on these evaluations, while permutation sampling is negligible. This bottleneck is shared by any Shapley-type method with an expensive utility: for N permutations and n players, direct GPASV needs Nn utility calls, the same as permutation-based Shapley value or PASV, and the MH sampling itself requires no utility calls. Note also that the total cost covers the full sweep over all 19 settings, not a single valuation. To scale to more players, one can run a sparser sweep grid or reduce the number of Monte Carlo permutations per setting.

Table 9: Realized utility-evaluation cost under the four combinations of subset reuse (cache) and permutation reuse (SNIS). "# of Permutations" is the total number of permutations sampled across the 19 regimes after SNIS thinning along the three sweeps (with the baseline $(0, 0)$ shared once); "# of Subsets" is the number of distinct prefix coalitions evaluated per MT-Bench prompt.

Scenario	Cache	SNIS	# of Permutations	# of Subsets
A. No reuse	✗	✗	19,000	380,000
B. Subset reuse only	✓	✗	19,000	78,220
C. Permutation reuse only	✗	✓	15,840	316,800
D. Both	✓	✓	15,840	70,511

Table 10: Permutations saved by SNIS along the three sweep traversals. "Without SNIS" is the 6 N_{MC} fresh permutations that would be drawn per sweep (6 non-baseline temperatures $\times$ $N_{\mathrm{MC}} = 1{,}000$); "with SNIS" is the realized $\sum N_{\mathrm{new}}$ under the rule above. The shared baseline $(0, 0)$ contributes a single $N_{\mathrm{MC}} = 1{,}000$ draw across all three sweeps and is excluded from this table.

Sweep	Without SNIS	With SNIS	Saved
α-only ($\beta = 0$)	6,000	4,007	1,993
β-only ($\alpha = 0$)	6,000	5,403	597
joint ($\alpha = \beta$)	6,000	5,430	570
Three-sweep total	18,000	14,840	3,160

H.4 Full Results

Section 6 showed results for representative regimes; this subsection reports the full results across all 19 settings generated by the three one-dimensional sweeps on (α, β), in three views: per-player value trajectories along each sweep (Figure 14), per-player rank under every (α, β) cell (Figure 15), and the top-8 ranked players at every individual regime (Figure 16).

Per-player value trajectories. Figure 14 plots $\psi_i(U_{\mathrm{ens}})$ for each of the 20 players as a function of the sweep temperature, one panel per sweep (α-only, β-only, joint $\alpha = \beta$). Solid lines are open-source models and dashed lines are paid. This view emphasizes the *trajectory* of each player as the priority strength is increased and is the curve-level companion to the group-sum view in Figure 7.

Per-player rank on the (α, β) grid. Figure 15 reports, for each of the 20 players, the rank of that player among all 20 models on every supported (α, β) cell of the 7×7 grid (the 19 cells along the three sweeps; off-support cells are masked white). Each per-player heatmap shares the same color scale: rank 1 (best) is dark and rank 20 (worst) is light. Per-panel titles are colored by group (paid in dark blue, open-source in burnt orange) for quick reading; the same color convention is used in the bar charts of Figure 16. This view is the cell-level dual of Figure 14: the trajectory in the latter corresponds to a row or column of cells in the former.

Top-8 ranking under every regime. Figure 6 in the main text visualizes the top-8-valued models under representative regimes. Figure 16 extends the same view to all 19 regimes, arranged as a 7×3 grid: rows index the temperature value (0 baseline together with $\{1, 2, 4, 8, 16, 32\}$) and columns index the sweep type. Walking down a column traces how the top-8 shifts as that sweep is strengthened from the baseline; walking across a row contrasts the three sweep types at a fixed temperature.

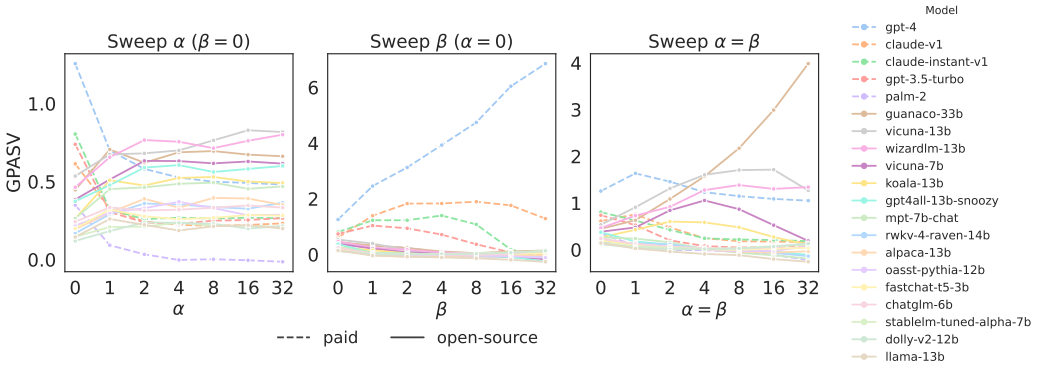


Figure 14: Per-player GPASV $\psi_i(U_{\mathrm{ens}})$ along the three sweeps, averaged over the 80 MT-Bench prompts. Left: α-only sweep ($\beta = 0$). Middle: β-only sweep ($\alpha = 0$). Right: joint sweep ($\alpha = \beta$). Solid lines are open-source models; dashed lines are paid (`gpt-4`, `gpt-3.5-turbo`, `claude-v1`, `claude-instant-v1`, `palm-2`).

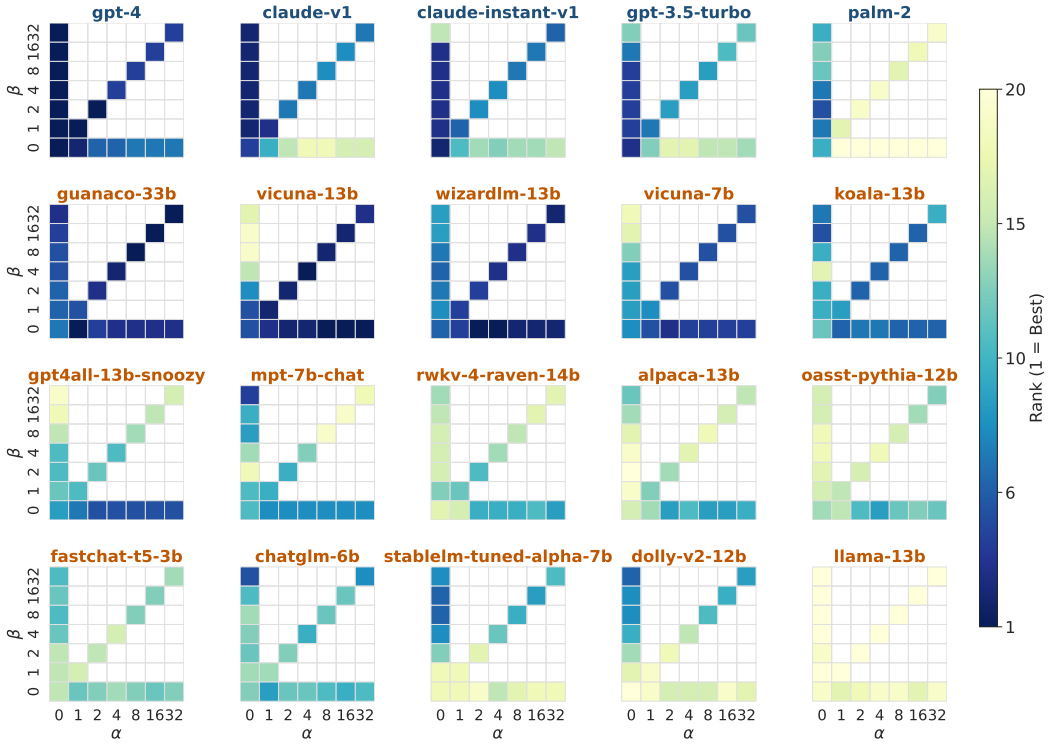


Figure 15: Per-player rank heatmaps over the supported (α, β) cells. Each subplot is one player; rows index β and columns index α (both on the grid $\{0, 1, 2, 4, 8, 16, 32\}$). Cells off the three sweep paths are masked. Per-panel title color encodes paid (blue) vs. open-source (orange), matching Figure 16.

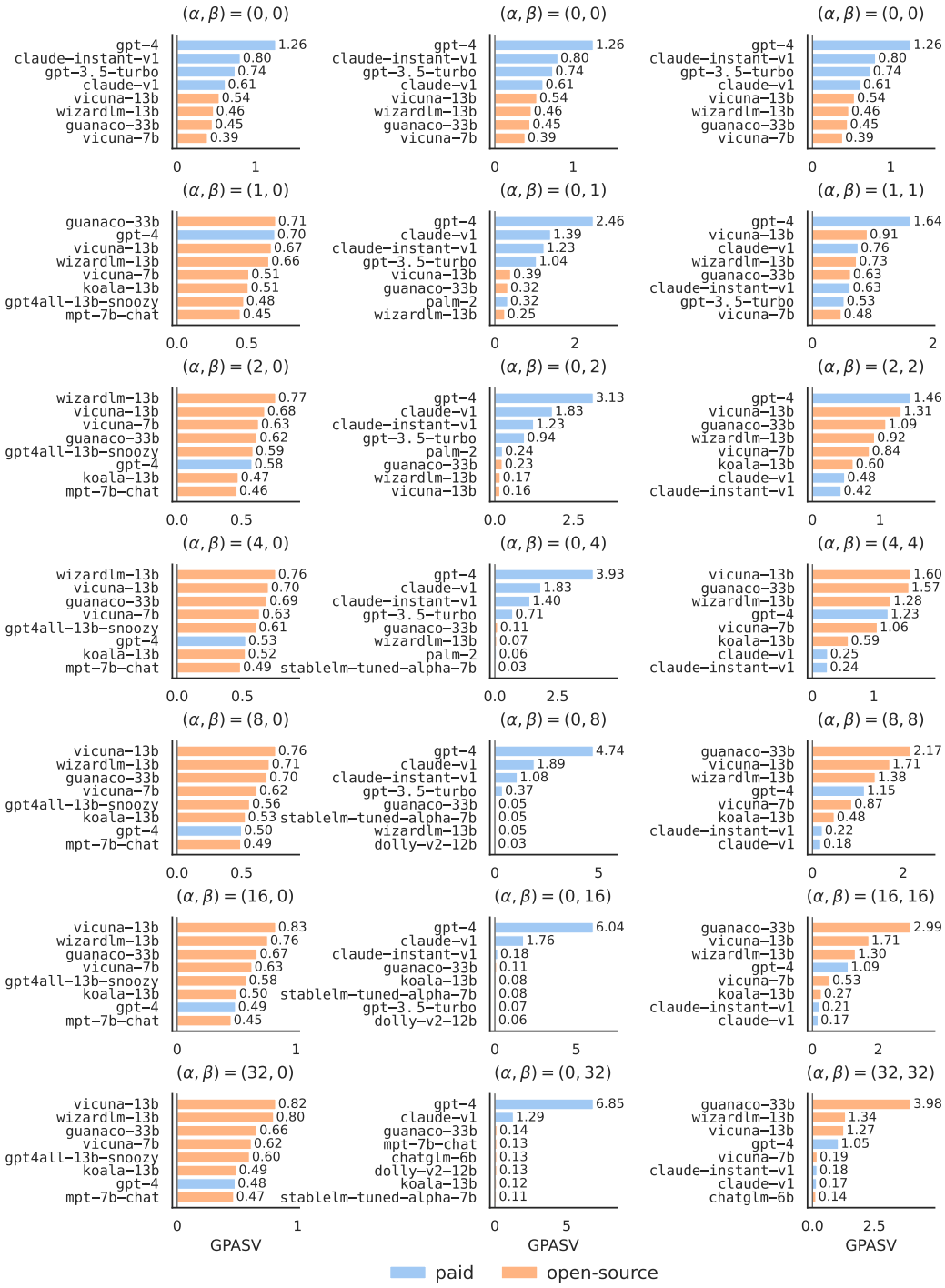


Figure 16: Top-8-valued models across all 19 priority regimes. Rows: temperature values (baseline $(0, 0)$ together with $\{1, 2, 4, 8, 16, 32\}$). Columns: α-only sweep ($\beta = 0$), β-only sweep ($\alpha = 0$), joint sweep ($\alpha = \beta$). Bar color encodes paid vs. open-source, matching the color scheme of Figure 6 in the main text.

H.5 Robustness to Priority-Graph Uncertainty and a Ranking-then-PASV Baseline

Graph bootstrap. The Chatbot Arena graph is estimated from finite comparisons. To assess how this affects GPASV, for each model pair we resample its win/loss/tie counts from a multinomial distribution with the observed proportions and the observed number of comparisons, and rebuild ω by the same rule as in Appendix H.1; we repeat this $B = 100$ times. For each bootstrap graph, we recompute GPASV at $\alpha = 0$ by self-normalized importance reweighting of a fixed pool of 6,000 permutations, sampled on the original graph at the six nonzero β's (1,000 each), with the cached utilities kept fixed. Table 11 reports the rank agreement between each bootstrap GPASV and the original-graph GPASV computed from the same pool. The rankings are stable for $\beta \leq 16$ (mean Kendall $\tau \geq 0.91$). The agreement decreases at $\beta = 32$, where fewer pooled permutations can be reused (low ESS), so we read $\beta = 16, 32$ as descriptive. These intervals capture priority-graph uncertainty only, conditional on the fixed utility and permutation pool.

Table 11: Graph-bootstrap stability of GPASV rankings ($\alpha = 0$, $B = 100$). Each cell reports the bootstrap mean and the $[5\%, 95\%]$ quantile interval.

β	ESS	Kendall τ	Spearman ρ
1	1369 [1328, 1412]	0.991 [0.979, 1.000]	0.999 [0.997, 1.000]
2	1770 [1679, 1848]	0.981 [0.958, 1.000]	0.996 [0.992, 1.000]
4	1345 [1154, 1520]	0.957 [0.926, 0.979]	0.991 [0.983, 0.997]
8	805 [481, 1088]	0.935 [0.884, 0.968]	0.985 [0.965, 0.995]
16	306 [112, 553]	0.912 [0.852, 0.958]	0.976 [0.949, 0.994]
32	80 [13, 209]	0.841 [0.662, 0.958]	0.939 [0.830, 0.991]

Ranking-then-PASV baseline. A natural alternative to GPASV is to first reduce the Arena comparisons to a total order and then apply PASV. We consider two such orders. Let c_{ij} be the number of comparisons between models i and j, and w_{ij} the points of i against j (one per win and one half per tie), so that $c_{ij} = w_{ij} + w_{ji}$. The Bradley–Terry ranking [3] orders models by decreasing $\widehat{s}_i$, where

$$\widehat{\mathbf{s}} = \underset{\sum_i s_i = 0}{\arg\max} \sum_{\{i,j\} \in \mathcal{E}} \Big[w_{ij} \log \sigma(s_i - s_j) + w_{ji} \log \sigma(s_j - s_i) \Big], \qquad \sigma(x) = \frac{1}{1 + e^{-x}},$$

and $\mathcal{E}$ is the set of retained pairs. The round-robin ranking orders models by decreasing average win share over all observed pairs, $r_i = \frac{1}{n-1} \sum_{j \neq i} w_{ij}/c_{ij}$. Each order defines a total-order DAG, on which PASV with $\lambda_i \equiv 1$ reduces to the marginal contributions along that single order. Since there is no ground-truth attribution, this comparison measures how well these baselines approximate GPASV, rather than which method is correct. Table 12 shows that the two baselines disagree with each other (Kendall $\tau = 0.418$), and both agree poorly with GPASV at $\beta = 1$ ($\tau \leq 0.174$); the agreement improves at $\beta = 32$ but remains moderate ($\tau \leq 0.459$). That is, forcing the cyclic graph into a total order changes the valuation, and the result also depends on which projection is chosen.

Table 12: Rank agreement between ranking-then-PASV baselines and GPASV at $\alpha = 0$. GPASV(α, β) is computed on the original graph from the same permutation pool as in Table 11.

Comparison	Kendall τ	Spearman ρ	Common top-5
BT–PASV vs. RR–PASV	0.418	0.398	2/5
BT–PASV vs. GPASV$(0, 1)$	0.174	0.271	3/5
RR–PASV vs. GPASV$(0, 1)$	0.164	0.230	2/5
BT–PASV vs. GPASV$(0, 32)$	0.459	0.590	3/5
RR–PASV vs. GPASV$(0, 32)$	0.364	0.554	3/5

H.6 Prompt Templates

Aggregator prompt template: 1st turn

```
[Instruction]
You are an impartial aggregator. Given a user prompt and multiple candidate
answers, produce one final answer.

Your aggregation is merge-leaning:
- Primarily combine, edit, and reorganize what is already in the candidates.
- Do not add new substantive content or external facts.

Rules:
1) Follow the user's requested format, constraints, and style exactly.
2) Use only information explicitly stated in the candidates. You may add
minimal connective wording for readability.
3) If candidates disagree and you cannot resolve it from the candidates, omit
the claim or mark it as uncertain.
4) Remove redundancy and irrelevant parts; choose the clearest phrasing among
candidates.
5) If JSON/code/strict format is required, keep it valid and do not introduce
new APIs/libraries not present in candidates.
6) Two-turn consistency: if the prompt is multi-turn, treat the turn-1
aggregated answer as the assistant's previous message when producing the turn-2
 aggregated answer.

After synthesizing, output ONLY the final aggregated answer, with no extra
commentary.

[Question]
{x1}

<|The Start of Candidates|>
[The Start of Candidate 1 Answer]
{candidate answer 1}
[The End of Candidate 1 Answer]

...

[The Start of Candidate K Answer]
{candidate answer K}
[The End of Candidate K Answer]
<|The End of Candidates|>
```

Aggregator prompt template: 2nd turn

```
[Instruction]
You are an impartial aggregator. Given a user prompt and multiple candidate
answers, produce one final answer.

Your aggregation is merge-leaning:
- Primarily combine, edit, and reorganize what is already in the candidates.
- Do not add new substantive content or external facts.

Rules:
1) Follow the user's requested format, constraints, and style exactly.
2) Use only information explicitly stated in the candidates. You may add
minimal connective wording for readability.
3) If candidates disagree and you cannot resolve it from the candidates, omit
the claim or mark it as uncertain.
4) Remove redundancy and irrelevant parts; choose the clearest phrasing among
candidates.
5) If JSON/code/strict format is required, keep it valid and do not introduce
new APIs/libraries not present in candidates.
6) Two-turn consistency: if the prompt is multi-turn, treat the turn-1
aggregated answer as the assistant's previous message when producing the turn-2
 aggregated answer.

After synthesizing, output ONLY the final aggregated answer, with no extra
commentary.

<|The Start of Previous Conversation with User|>
### User:
{x1}

### Assistant:
{y1S}

### User:
{x2}
<|The End of Previous Conversation with User|>

<|The Start of Candidates|>
[The Start of Candidate 1 Answer]
{candidate answer 1}
[The End of Candidate 1 Answer]

...

[The Start of Candidate K Answer]
{candidate answer K}
[The End of Candidate K Answer]
<|The End of Candidates|>
```

Judge prompt template without reference: 1st turn

```
[Instruction]
You are an impartial judge evaluating the quality of an assistant's response.
Evaluate the response based on the following criteria:
- helpfulness
- relevance
- accuracy
- clarity
- completeness

Think carefully about the response before rating it.
Provide at most three short sentences summarizing the main reason for your
score.
Then, on a new final line, output the rating strictly in the format "[[rating
]]", where rating is an integer from 1 to 10, for example: "[[5]]".
Do not use any other rating format.
Do not output anything after the final rating line.

[Question]
{x1}

[The Start of Assistant's Answer]
{y1S}
[The End of Assistant's Answer]
```

Judge prompt template without reference: 2nd turn

```
[Instruction]
You are an impartial judge evaluating the quality of an assistant's response.
Evaluate the response based on the following criteria:
- helpfulness
- relevance
- accuracy
- clarity
- completeness

Think carefully about the response before rating it.
Provide at most three short sentences summarizing the main reason for your
score.
Then, on a new final line, output the rating strictly in the format "[[rating
]]", where rating is an integer from 1 to 10, for example: "[[5]]".
Do not use any other rating format.
Do not output anything after the final rating line.

<|The Start of Assistant A's Conversation with User|>
### User:
{x1}

### Assistant A:
{y1S}

### User:
{x2}

### Assistant A:
{y2S}
<|The End of Assistant A's Conversation with User|>
```

Judge prompt template with reference: 1st turn

```
[Instruction]
You are an impartial judge evaluating the quality of an assistant's response
using a reference answer.
Evaluate the response by comparing it with the reference answer based on the
following criteria:
- correctness
- helpfulness
- relevance
- clarity
- completeness

Think carefully about the response before rating it.
If the assistant's response misses important points or contains mistakes
compared with the reference answer, take that into account.
Provide at most three short sentences summarizing the main reason for your
score.
Then, on a new final line, output the rating strictly in the format "[[rating
]]", where rating is an integer from 1 to 10, for example: "[[5]]".
Do not use any other rating format.
Do not output anything after the final rating line.

[Question]
{x1}

[The Start of Reference Answer]
{r1}
[The End of Reference Answer]

[The Start of Assistant's Answer]
{y1S}
[The End of Assistant's Answer]
```

Judge prompt template with reference: 2nd turn

```
[Instruction]
You are an impartial judge evaluating the quality of an assistant's response
using a reference answer.
Evaluate the response by comparing it with the reference answer based on the
following criteria:
- correctness
- helpfulness
- relevance
- clarity
- completeness

Think carefully about the response before rating it.
If the assistant's response misses important points or contains mistakes
compared with the reference answer, take that into account.
Provide at most three short sentences summarizing the main reason for your
score.
Then, on a new final line, output the rating strictly in the format "[[rating
]]", where rating is an integer from 1 to 10, for example: "[[5]]".
Do not use any other rating format.
Do not output anything after the final rating line.

<|The Start of Reference Answer|>
### User:
{x1}

### Reference answer:
{r1}

### User:
{x2}

### Reference answer:
{r2}
<|The End of Reference Answer|>

<|The Start of Assistant A's Conversation with User|>
### User:
{x1}

### Assistant A:
{y1S}

### User:
{x2}

### Assistant A:
{y2S}
<|The End of Assistant A's Conversation with User|>
```

I Licenses

Here we list the licenses for the public assets used in this paper. For each asset, we also list their corresponding citation entry in our reference list.

Asset	Source/Version	License/Terms	Citation
MT-Bench benchmark assets and released model responses	LMSYS/FastChat MT-Bench assets; 80 two-turn prompts, reference answers, evaluation protocol, and released model answers for the evaluated overlap models	FastChat repository assets are released under the Apache License 2.0. We use the released benchmark responses only as fixed candidate answers for evaluation; where applicable, use of model outputs also follows the corresponding model/provider terms.	[46]
Chatbot Arena pairwise preference data	`lmsys/chatbot_arena_conversations` public release; pairwise human-preference records used to construct the priority graph	Dataset terms state that user prompts are licensed under CC-BY-4.0 and model outputs under CC-BY-NC-4.0; users must follow applicable laws, model-specific terms, and not attempt re-identification.	[7, 46]
Qwen3.5-35B-A3B-FP8	`Qwen/Qwen3.5-35B-A3B-FP8`; FP8-quantized model weights and configuration files used for the aggregator and judge	Apache License 2.0. We use the model only for local inference in the aggregator–judge pipeline and do not fine-tune it.	[37]
vLLM inference engine	vLLM open-source inference-serving library used to serve Qwen3.5-35B-A3B-FP8 locally	Apache License 2.0. We use vLLM only as the local inference backend.	[24]

Table 13: Existing public assets used in this paper and their licenses and citations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflected the contents of the paper, and the conclusions are supported by theory and numerical experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section A in Appendix for details.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Appendix contains complete proofs to all theoretical results in the main paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide full details for reproducibility, including publicly released computer code and appendix sections that contain experimental details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Data: we use public data; computer code: we release our computer code at https://github.com/KiljaeL/GPASV.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All details are in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes] (when applicable)

   Justification: We reported error bars whenever computational cost is affordable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All details are in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We adhere to Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed that our work is a reminder for data collectors that some important metadata should be collected. Our work does not have clearly identifiable negative societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This question is not relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We only use public data sets and have been conforming to their license terms. For more details, see Appendix I.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This question is not relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This question is not relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This question is not relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: After carefully reading the LLM policy of NeurIPS 2026, we believe no LLM-related declaration is needed for this paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.