Human review packet: GGSG19TopThree

Generated from byte-pinned source excerpts, the expanded PaperInterface specifications, and hash-bound raw-source-to-Spec screening records.

Paper: GGSG19TopThree
Paper id: GGSG19TopThree
Source version: arXiv:1906.08160, 2019; HCOMP 2019 source bundle
Source record: audit/paper_statement_map.json
Rows: 15
Generated: 2026-09-07
Source URL: official source

How to use this packet

For each source claim, compare the exact source excerpts with the expanded PaperInterface specification. Mark up this PDF or fill the blank reviewer box in the TeX source; return the annotated file for reconciliation into the paper-local human-review log.

Open the interactive dashboard

The interactive dashboard is an optional alternative to using this PDF; it presents the same review surface rather than an additional review requirement. After forking and cloning (or pulling) AppliedModelingLib, run the following from the repository root. The cache command is needed only the first time in a new clone.

```
cd <your-repository-checkout>
lake exe cache get
python3 scripts/review_dashboard.py --paper GGSG19TopThree --serve
```

Open http://127.0.0.1:8765/ in a browser and keep that terminal running while reviewing. The dashboard records saved annotations in this paper's local reviewer-owned review record.

Contents

- Governing Lean declarations (134; supporting context)
- Semantic library prerequisites (1; not paper claims)
  - AppliedModelingLib.SocialChoice.Ranking.kApprovalScore
- Paper-specific semantic prerequisites (19; not paper claims)
  - GGSG19TopThree.ProofBridge.iidUniformTieCandidateAverageScore
  - GGSG19TopThree.ProofBridge.kRankingScore
  - GGSG19TopThree.ProofBridge.paper_definition_design_invariant
  - GGSG19TopThree.ProofBridge.paper_definition_large_deviation_rate
  - GGSG19TopThree.ProofBridge.paper_definition_rate_optimal
  - GGSG19TopThree.ReasonablePrefixWeights
  - GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain
  - GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain
  - GGSG19TopThree.assumption_nontrivial_k_approval_cutoffs
  - GGSG19TopThree.assumption_pairwise_approval_ternary_gap_domain
  - GGSG19TopThree.assumption_randomized_mechanism_probability_weights
  - GGSG19TopThree.iidUniformTieTieredCorrect
  - GGSG19TopThree.mallowsPairDPTabulationRun
  - GGSG19TopThree.model_arbitrary_ordered_tier_goal
  - GGSG19TopThree.model_finite_iid_strict_rankings
  - GGSG19TopThree.model_mallows_ranking_mass_and_parameter_domain
  - GGSG19TopThree.model_randomized_scoring_mechanism
  - GGSG19TopThree.rankingPrefixScore
  - GGSG19TopThree.rankingUniformTieSignalLaw
- Source claims (15)
  - source_proposition1_thm_consistency_tieredSpec
  - source_proposition2_thm_pairwiselearning_finite_supportSpec

- source_proposition3_lem_pairwiselearning_approval_finite_ternarySpec
- source_proposition4_thm_goal_learning_exact_minimum_rateSpec
- source_proposition4_thm_goal_learning_finite_sample_M_sq_boundSpec
- source_theorem1_lem_randomizebetterscoring_arbitrary_goalSpec
- source_theorem2_lem_randomizenotbetterapproval_pairwiseSpec
- source_corollary_lem_mallowsnorando_zero_noiseSpec
- source_corollary_lem_mallowsnorandoSpec
- source_theorem_lem_randomizebetterapproval_w_selection_constructedSpec
- source_theorem_lem_mallowsnotWK_counterexampleSpec
- source_proof_mallows_common_pivotal_pairSpec
- source_mallows_repeated_insertion_probability_formulaSpec
- source_proof_mallows_four_candidate_joint_location_matrixSpec
- source_example_durham_ward1_randomized_approval_improves_static3_static4Spec

Governing Lean declarations

These exact graph-bound declaration bodies are retained by the displayed specifications or reviewed prerequisites. They are supporting context and do not add source-result rows or semantic judgments.

AppliedModelingLib.Probability.finiteIidScoreSum

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → {ι : Type u_2} → [Fintype ι] → (α → ℝ) → (ι → α) → ℝ
```

value:
```
fun {α : Type u_1} {ι : Type u_2} [Fintype ι] (score : α → ℝ) (sample : ι → α) => ∑ i : ι, score (sample i)
```

AppliedModelingLib.Probability.finiteMGF

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℝ
```

value:
```
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (z : ℝ) => ∑ a : α, (μ a).toReal * Real.exp (z * score a)
```

AppliedModelingLib.Probability.finiteLogMGF

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℝ
```

value:
```
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (z : ℝ) =>
  Real.log (AppliedModelingLib.Probability.finiteMGF μ score z)
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteMGF

AppliedModelingLib.Probability.finiteChernoffRate

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ
```

value:
```
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) =>
  -sInf (Set.range fun (z : ℝ) => AppliedModelingLib.Probability.finiteLogMGF μ score z)
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteLogMGF

AppliedModelingLib.Probability.finiteLegendreValue

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℝ → ℝ
```

value:
```
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (a z : ℝ) =>
  z * a - AppliedModelingLib.Probability.finiteLogMGF μ score z
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteLogMGF

AppliedModelingLib.Probability.finiteRateFunctionTop

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → WithTop ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (a : ℝ) =>
  sSup (Set.range fun (z : ℝ) => ↑(AppliedModelingLib.Probability.finiteLegendreValue μ score a z))

Direct retained dependencies

- AppliedModelingLib.Probability.finiteLegendreValue

AppliedModelingLib.Probability.finiteScoreGapChernoffRate

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → (α → ℝ) → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (hiScore loScore : α → ℝ) =>
  AppliedModelingLib.Probability.finiteChernoffRate μ fun (a : α) => hiScore a - loScore a

Direct retained dependencies

- AppliedModelingLib.Probability.finiteChernoffRate

AppliedModelingLib.Probability.logDecay

Declaration location: reusable library

Lean declaration body

type:
(ℕ → ℝ) → ℕ → ℝ

value:
fun (p : ℕ → ℝ) (n : ℕ) => -(↑n)⁻¹ * Real.log (p n)

AppliedModelingLib.Probability.HasExponentialRate

Declaration location: reusable library

Lean declaration body

type:
(ℕ → ℝ) → ℝ → Prop

value:
fun (p : ℕ → ℝ) (rate : ℝ) => Filter.Tendsto (AppliedModelingLib.Probability.logDecay p) Filter.atTop (nhds rate)

Direct retained dependencies

- AppliedModelingLib.Probability.logDecay

AppliedModelingLib.Probability.ExponentialRateCertificate

Declaration location: reusable library

Lean declaration body

field eventually_pos:
∀ {p : ℕ → ℝ} {rate : ℝ},
  AppliedModelingLib.Probability.ExponentialRateCertificate p rate → ∀ᶠ (n : ℕ) in Filter.atTop, 0 < p n

field has_rate:
∀ {p : ℕ → ℝ} {rate : ℝ},
  AppliedModelingLib.Probability.ExponentialRateCertificate p rate →
    AppliedModelingLib.Probability.HasExponentialRate p rate

Direct retained dependencies

- AppliedModelingLib.Probability.HasExponentialRate

AppliedModelingLib.Probability.HasExtendedExponentialRate

Declaration location: reusable library

Lean declaration body

type:
(ℕ → ℝ) → WithTop ℝ → Prop

constructor AppliedModelingLib.Probability.HasExtendedExponentialRate.finite:
∀ {p : ℕ → ℝ} {rate : ℝ},
  AppliedModelingLib.Probability.HasExponentialRate p rate →
    AppliedModelingLib.Probability.HasExtendedExponentialRate p ↑rate

constructor AppliedModelingLib.Probability.HasExtendedExponentialRate.infinite:
∀ {p : ℕ → ℝ}, (∀ᶠ (n : ℕ) in Filter.atTop, p n = 0) → AppliedModelingLib.Probability.HasExtendedExponentialRate p ⊤

Direct retained dependencies

- AppliedModelingLib.Probability.HasExponentialRate

AppliedModelingLib.Probability.ternaryGapClosedChernoffRate

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pUp pDown : ℝ) => -Real.log (2 * √(pUp * pDown) + 1 - pUp - pDown)

AppliedModelingLib.SocialChoice.Ranking.Candidate

Declaration location: reusable library

Lean declaration body

type:
ℕ → Type

value:
fun (n : ℕ) => Fin (n + 2)

AppliedModelingLib.SocialChoice.Ranking.Ranking

Declaration location: reusable library

Lean declaration body

type:
ℕ → Type

value:
fun (n : ℕ) => Equiv.Perm (AppliedModelingLib.SocialChoice.Ranking.Candidate n)

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate

AppliedModelingLib.SocialChoice.Ranking.firstChoice

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n

value:
fun {n : ℕ} (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) => π 0
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking

AppliedModelingLib.SocialChoice.Ranking.instMeasurableSpaceRanking

Declaration location: reusable library

Lean declaration body

```
type:
(n : ℕ) → MeasurableSpace (AppliedModelingLib.SocialChoice.Ranking.Ranking n)

value:
fun (n : ℕ) => ⊤
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Ranking

AppliedModelingLib.SocialChoice.Ranking.rankOf

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.Ranking n →
    AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Candidate n

value:
fun {n : ℕ} (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  (Equiv.symm π) c
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking

AppliedModelingLib.SocialChoice.Ranking.approvedByK

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  ℕ → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → Prop

value:
fun {n : ℕ} (K : ℕ) (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  ↑(AppliedModelingLib.SocialChoice.Ranking.rankOf π c) < K
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf

AppliedModelingLib.SocialChoice.Ranking.instDecidableApprovedByK

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  (K : ℕ) →
    (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
      (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) →
        Decidable (AppliedModelingLib.SocialChoice.Ranking.approvedByK K π c)

value:
fun {n : ℕ} (K : ℕ) (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  id inferInstance
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.approvedByK
- AppliedModelingLib.SocialChoice.Ranking.rankOf

AppliedModelingLib.SocialChoice.Ranking.instDecidablePredApprovedByK

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  (K : ℕ) →
    (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) →
      (a : AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
        Decidable
          ((fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
              AppliedModelingLib.SocialChoice.Ranking.approvedByK K π c)
            a)

value:
fun {n : ℕ} (K : ℕ) (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n)
    (a : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
  inferInstance
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.approvedByK
- AppliedModelingLib.SocialChoice.Ranking.instDecidableApprovedByK

AppliedModelingLib.SocialChoice.Ranking.invertedPair

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.Ranking n →
    AppliedModelingLib.SocialChoice.Ranking.Ranking n →
      AppliedModelingLib.SocialChoice.Ranking.Candidate n × AppliedModelingLib.SocialChoice.Ranking.Candidate n → Prop

value:
fun {n : ℕ} (ρ π : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (ab : AppliedModelingLib.SocialChoice.Ranking.Candidate n × AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  AppliedModelingLib.SocialChoice.Ranking.rankOf ρ ab.1 < AppliedModelingLib.SocialChoice.Ranking.rankOf ρ ab.2 ∧
    AppliedModelingLib.SocialChoice.Ranking.rankOf π ab.2 < AppliedModelingLib.SocialChoice.Ranking.rankOf π ab.1
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf

AppliedModelingLib.SocialChoice.Ranking.inversionFinset

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.Ranking n →
    AppliedModelingLib.SocialChoice.Ranking.Ranking n →
      Finset (AppliedModelingLib.SocialChoice.Ranking.Candidate n × AppliedModelingLib.SocialChoice.Ranking.Candidate n)

value:
fun {n : ℕ} (ρ π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
  {ab : AppliedModelingLib.SocialChoice.Ranking.Candidate n × AppliedModelingLib.SocialChoice.Ranking.Candidate n |
    AppliedModelingLib.SocialChoice.Ranking.invertedPair ρ π ab}
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.invertedPair

AppliedModelingLib.SocialChoice.Ranking.kendallTau

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℕ

value:
fun {n : ℕ} (ρ π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
  (AppliedModelingLib.SocialChoice.Ranking.inversionFinset ρ π).card
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.inversionFinset

AppliedModelingLib.SocialChoice.Ranking.mallowsWeight

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} → ℝ → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ

value:
fun {n : ℕ} (q : ℝ) (ρ π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
  q ^ AppliedModelingLib.SocialChoice.Ranking.kendallTau ρ π
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.kendallTau

AppliedModelingLib.SocialChoice.Ranking.mallowsPartition

Declaration location: reusable library

Lean declaration body

type:
{n : ℕ} → ℝ → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ

value:
fun {n : ℕ} (q : ℝ) (ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
  ∑ π : AppliedModelingLib.SocialChoice.Ranking.Ranking n, AppliedModelingLib.SocialChoice.Ranking.mallowsWeight q ρ π

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.mallowsWeight

AppliedModelingLib.SocialChoice.Ranking.MallowsSpec

Declaration location: reusable library

Lean declaration body

field center:
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n → AppliedModelingLib.SocialChoice.Ranking.Ranking n

field q:
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n → ℝ

field law:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n → PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)

field partition:
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n → ℝ

field q_pos:
∀ {n : ℕ} (self : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n), 0 < self.q

field partition_pos:
∀ {n : ℕ} (self : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n), 0 < self.partition

field partition_eq_sum:
∀ {n : ℕ} (self : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n),
  self.partition = AppliedModelingLib.SocialChoice.Ranking.mallowsPartition self.q self.center

field law_apply_toReal:
∀ {n : ℕ} (self : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
  (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n),
  (self.law π).toReal = AppliedModelingLib.SocialChoice.Ranking.mallowsWeight self.q self.center π / self.partition

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.mallowsPartition
- AppliedModelingLib.SocialChoice.Ranking.mallowsWeight

AppliedModelingLib.SocialChoice.Ranking.mallowsPMF

Declaration location: reusable library

Lean declaration body

type:
{n : ℕ} →
  (q : ℝ) →
    AppliedModelingLib.SocialChoice.Ranking.Ranking n → 0 < q → PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)

value:
fun {n : ℕ} (q : ℝ) (ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n) (hq : 0 < q) =>
  PMF.ofFintype
    (fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
      ENNReal.ofReal
        (AppliedModelingLib.SocialChoice.Ranking.mallowsWeight q ρ π /
          AppliedModelingLib.SocialChoice.Ranking.mallowsPartition q ρ))
    (AppliedModelingLib.SocialChoice.Ranking.mallowsPMF._proof_1 q ρ hq)

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.mallowsPartition
- AppliedModelingLib.SocialChoice.Ranking.mallowsWeight

AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.Ranking n →
    (q : ℝ) → 0 < q → AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n

value:
fun {n : ℕ} (ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n) (q : ℝ) (hq : 0 < q) =>
  { center := ρ, q := q, law := AppliedModelingLib.SocialChoice.Ranking.mallowsPMF q ρ hq,
    partition := AppliedModelingLib.SocialChoice.Ranking.mallowsPartition q ρ, q_pos := hq,
    partition_pos := AppliedModelingLib.SocialChoice.Ranking.mallowsPartition_pos hq ρ,
    partition_eq_sum := AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ._proof_1 ρ q,
    law_apply_toReal := fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
      AppliedModelingLib.SocialChoice.Ranking.mallowsPMF_apply_toReal hq ρ π }
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.mallowsPMF
- AppliedModelingLib.SocialChoice.Ranking.mallowsPartition
- AppliedModelingLib.SocialChoice.Ranking.mallowsWeight

AppliedModelingLib.SocialChoice.Ranking.secondChoice

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n

value:
fun {n : ℕ} (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) => π 1
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking

AppliedModelingLib.finitePMFIidPathMeasure

Declaration location: reusable library

Lean declaration body

```
type:
{Signal : Type u_1} →
  [inst : MeasurableSpace Signal] →
    [Fintype Signal] → [DecidableEq Signal] → PMF Signal → MeasureTheory.Measure (ℕ → Signal)

value:
fun {Signal : Type u_1} [MeasurableSpace Signal] [Fintype Signal] [DecidableEq Signal] (law : PMF Signal) =>
  MeasureTheory.Measure.infinitePi fun (x : ℕ) => law.toMeasure
```

AppliedModelingLib.finiteWeightedPMF

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [inst : Fintype α] → (weight : α → ℝ) → (∀ (a : α), 0 ≤ weight a) → 0 < ∑ a : α, weight a → PMF α

value:
fun {α : Type u_1} [Fintype α] (weight : α → ℝ) (hweight_nonneg : ∀ (a : α), 0 ≤ weight a)
    (htotal_pos : 0 < ∑ a : α, weight a) =>
  PMF.ofFintype (fun (a : α) => ENNReal.ofReal (weight a / ∑ b : α, weight b))
    (AppliedModelingLib.finiteWeightedPMF._proof_1 weight hweight_nonneg htotal_pos)

AppliedModelingLib.pmfExp

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (f : α → ℝ) => ∑ a : α, (μ a).toReal * f a

AppliedModelingLib.pmfProb

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (p : α → Prop) → [DecidablePred p] → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (p : α → Prop) [DecidablePred p] =>
  AppliedModelingLib.pmfExp μ fun (a : α) => if p a then 1 else 0

Direct retained dependencies

- AppliedModelingLib.pmfExp

AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.firstSecondChoiceProb

Declaration location: reusable library

Lean declaration body

type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n →
    AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (c d : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  AppliedModelingLib.pmfProb M.law fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
    c = AppliedModelingLib.SocialChoice.Ranking.firstChoice π ∧
      d = AppliedModelingLib.SocialChoice.Ranking.secondChoice π

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.firstChoice
- AppliedModelingLib.SocialChoice.Ranking.secondChoice
- AppliedModelingLib.pmfProb

AppliedModelingLib.SocialChoice.Ranking.firstChoiceProb

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (μ : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n))
    (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  AppliedModelingLib.pmfProb μ fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
    c = AppliedModelingLib.SocialChoice.Ranking.firstChoice π
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.firstChoice
- AppliedModelingLib.pmfProb

AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    ℕ → AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (μ : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)) (K : ℕ)
    (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  AppliedModelingLib.pmfProb μ fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
    AppliedModelingLib.SocialChoice.Ranking.approvedByK K π lo ∧
      ¬AppliedModelingLib.SocialChoice.Ranking.approvedByK K π hi
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.approvedByK
- AppliedModelingLib.SocialChoice.Ranking.instDecidablePredApprovedByK
- AppliedModelingLib.pmfProb

AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb

Declaration location: reusable library

Lean declaration body

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    ℕ → AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (μ : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)) (K : ℕ)
    (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  AppliedModelingLib.pmfProb μ fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
    AppliedModelingLib.SocialChoice.Ranking.approvedByK K π hi ∧
      ¬AppliedModelingLib.SocialChoice.Ranking.approvedByK K π lo
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.approvedByK
- AppliedModelingLib.SocialChoice.Ranking.instDecidablePredApprovedByK
- AppliedModelingLib.pmfProb

AppliedModelingLib.pmfProd

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} → {β : Type u_2} → [Fintype α] → [Fintype β] → PMF α → PMF β → PMF (α × β)

value:
fun {α : Type u_1} {β : Type u_2} [Fintype α] [Fintype β] (μ : PMF α) (ν : PMF β) =>
  PMF.ofFintype (fun (p : α × β) => μ p.1 * ν p.2) (AppliedModelingLib.pmfProd._proof_1 μ ν)
```

AppliedModelingLib.pmfProduct

Declaration location: reusable library

Lean declaration body

```
type:
(ι : Type u_1) → (α : Type u_2) → [Fintype ι] → [DecidableEq ι] → [Fintype α] → PMF α → PMF (ι → α)

value:
fun (ι : Type u_1) (α : Type u_2) [Fintype ι] [DecidableEq ι] [Fintype α] (μ : PMF α) =>
  PMF.ofFintype (fun (f : ι → α) => ∏ i : ι, μ (f i)) (AppliedModelingLib.pmfProduct._proof_1 ι α μ)
```

AppliedModelingLib.Probability.finiteIidScoreLeftTailProb

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℕ → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (threshold : ℝ) (n : ℕ) =>
  AppliedModelingLib.pmfProb (AppliedModelingLib.pmfProduct (Fin n) α μ) fun (sample : Fin n → α) =>
    AppliedModelingLib.Probability.finiteIidScoreSum score sample ≤ threshold
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteIidScoreSum
- AppliedModelingLib.pmfProb
- AppliedModelingLib.pmfProduct

AppliedModelingLib.Probability.finiteIidScoreGapLeftTailProb

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → (α → ℝ) → ℕ → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (hiScore loScore : α → ℝ) (n : ℕ) =>
  AppliedModelingLib.Probability.finiteIidScoreLeftTailProb μ (fun (a : α) => hiScore a - loScore a) 0 n
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteIidScoreLeftTailProb

GGSG19TopThree.ConstructedWSelectionCandidate

Declaration location: paper-local

Lean declaration body

type:
Type

constructor GGSG19TopThree.ConstructedWSelectionCandidate.winner:
GGSG19TopThree.ConstructedWSelectionCandidate

constructor GGSG19TopThree.ConstructedWSelectionCandidate.loserI:
GGSG19TopThree.ConstructedWSelectionCandidate

constructor GGSG19TopThree.ConstructedWSelectionCandidate.loserJ:
GGSG19TopThree.ConstructedWSelectionCandidate

GGSG19TopThree.ConstructedWSelectionPair

Declaration location: paper-local

Lean declaration body

type:
Type

constructor GGSG19TopThree.ConstructedWSelectionPair.hi:
GGSG19TopThree.ConstructedWSelectionPair

constructor GGSG19TopThree.ConstructedWSelectionPair.hj:
GGSG19TopThree.ConstructedWSelectionPair

GGSG19TopThree.ConstructedWSelectionRanking

Declaration location: paper-local

Lean declaration body

type:
Type

constructor GGSG19TopThree.ConstructedWSelectionRanking.hij:
GGSG19TopThree.ConstructedWSelectionRanking

constructor GGSG19TopThree.ConstructedWSelectionRanking.hji:
GGSG19TopThree.ConstructedWSelectionRanking

constructor GGSG19TopThree.ConstructedWSelectionRanking.ihj:
GGSG19TopThree.ConstructedWSelectionRanking

constructor GGSG19TopThree.ConstructedWSelectionRanking.ijh:
GGSG19TopThree.ConstructedWSelectionRanking

constructor GGSG19TopThree.ConstructedWSelectionRanking.jhi:
GGSG19TopThree.ConstructedWSelectionRanking

constructor GGSG19TopThree.ConstructedWSelectionRanking.jih:
GGSG19TopThree.ConstructedWSelectionRanking

GGSG19TopThree.ConstructedWSelectionRanking1

Declaration location: paper-local

Lean declaration body

type:
Type

value:
AppliedModelingLib.SocialChoice.Ranking.Ranking 1

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Ranking

GGSG19TopThree.ConstructedWSelectionRanking1Candidate

Declaration location: paper-local

Lean declaration body

type:
Type

value:
AppliedModelingLib.SocialChoice.Ranking.Candidate 1

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate

GGSG19TopThree.DurhamWard1Pair

Declaration location: paper-local

Lean declaration body

type:
Type

constructor GGSG19TopThree.DurhamWard1Pair.hj:
GGSG19TopThree.DurhamWard1Pair

constructor GGSG19TopThree.DurhamWard1Pair.ij:
GGSG19TopThree.DurhamWard1Pair

GGSG19TopThree.MallowsPairPositionState

Declaration location: paper-local

Lean declaration body

type:
ℕ → Type

value:
fun (N : ℕ) => Option (Fin N) × Option (Fin N)

GGSG19TopThree.MallowsPairDPTabulationResult

Declaration location: paper-local

Lean declaration body

field table:
{N : ℕ} → GGSG19TopThree.MallowsPairDPTabulationResult N → GGSG19TopThree.MallowsPairPositionState N → ℝ

field scalarOperations:
{N : ℕ} → GGSG19TopThree.MallowsPairDPTabulationResult N → ℕ

Direct retained dependencies

- GGSG19TopThree.MallowsPairPositionState

GGSG19TopThree.MallowsRepeatedInsertionModel

Declaration location: paper-local

Lean declaration body

field q:
{N : ℕ} → GGSG19TopThree.MallowsRepeatedInsertionModel N → ℝ

field q_nonneg:
∀ {N : ℕ} (self : GGSG19TopThree.MallowsRepeatedInsertionModel N), 0 ≤ self.q

field q_le_one:
∀ {N : ℕ} (self : GGSG19TopThree.MallowsRepeatedInsertionModel N), self.q ≤ 1

field insertion:
{N : ℕ} → GGSG19TopThree.MallowsRepeatedInsertionModel N → Fin N → PMF (Fin N)

field insertion_apply_toReal:
∀ {N : ℕ} (self : GGSG19TopThree.MallowsRepeatedInsertionModel N) (stage inserted : Fin N),
((self.insertion stage) inserted).toReal =
if ↑inserted ≤ ↑stage then self.q ^ (↑stage - ↑inserted) / ∑ r ∈ Finset.range (↑stage + 1), self.q ^ (↑stage - r)
else 0

GGSG19TopThree.ProofBridge.paper_definition_design_invariant_for

Declaration location: paper-local

Lean declaration body

type:
```
{Ω : Type u_1} →
  {Rule : Type u_2} →
    {Outcome : Type u_3} →
      [inst : MeasurableSpace Ω] → MeasureTheory.Measure Ω → (Rule → Prop) → (Rule → ℕ → Ω → Outcome) → Outcome → Prop
```

value:
```
fun {Ω : Type u_1} {Rule : Type u_2} {Outcome : Type u_3} [MeasurableSpace Ω] (μ : MeasureTheory.Measure Ω)
    (reasonable : Rule → Prop) (outcome : Rule → ℕ → Ω → Outcome) (Ostar : Outcome) =>
  ∀ (β : Rule), reasonable β → ∀ᵐ (ω : Ω) ∂μ, ∀ᶠ (N : ℕ) in Filter.atTop, outcome β N ω = Ostar
```

GGSG19TopThree.RankingProperPrefixCut

Declaration location: paper-local

Lean declaration body

type:
```
ℕ → Type
```

value:
```
fun (n : ℕ) => Fin (n + 1)
```

GGSG19TopThree.StrictTopPrefixDominance

Declaration location: paper-local

Lean declaration body

type:
```
{Cut : Type u_1} → {Candidate : Type u_2} → (Candidate → Cut → ℝ) → Candidate → Candidate → Prop
```

value:
```
fun {Cut : Type u_1} {Candidate : Type u_2} (topPrefixProb : Candidate → Cut → ℝ) (hi lo : Candidate) =>
  ∀ (k : Cut), topPrefixProb lo k < topPrefixProb hi k
```

GGSG19TopThree.StrictTopPrefixDominanceOn

Declaration location: paper-local

Lean declaration body

type:
```
{Cut : Type u_1} → {Candidate : Type u_2} → (Candidate → Cut → ℝ) → (Candidate → Candidate → Prop) → Prop
```

value:
```
fun {Cut : Type u_1} {Candidate : Type u_2} (topPrefixProb : Candidate → Cut → ℝ)
    (crossTier : Candidate → Candidate → Prop) =>
  ∀ (hi lo : Candidate), crossTier hi lo → GGSG19TopThree.StrictTopPrefixDominance topPrefixProb hi lo
```

Direct retained dependencies

- GGSG19TopThree.StrictTopPrefixDominance

GGSG19TopThree.TopWSelectionPair

Declaration location: paper-local

Lean declaration body

type:
```
{n : ℕ} → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → Type
```

value:
```
fun {n : ℕ} (ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  { pair : AppliedModelingLib.SocialChoice.Ranking.Candidate n × AppliedModelingLib.SocialChoice.Ranking.Candidate n //
    AppliedModelingLib.SocialChoice.Ranking.rankOf ρ pair.1 < W ∧
      W ≤ AppliedModelingLib.SocialChoice.Ranking.rankOf ρ pair.2 }
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf

GGSG19TopThree.TopWSelectionPair.hi

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n} →
    {W : AppliedModelingLib.SocialChoice.Ranking.Candidate n} →
      GGSG19TopThree.TopWSelectionPair ρ W → AppliedModelingLib.SocialChoice.Ranking.Candidate n

value:
fun {n : ℕ} {ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n}
    {W : AppliedModelingLib.SocialChoice.Ranking.Candidate n} (pair : GGSG19TopThree.TopWSelectionPair ρ W) =>
  (↑pair).1
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.TopWSelectionPair

GGSG19TopThree.TopWSelectionPair.instFintype

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n} →
    {W : AppliedModelingLib.SocialChoice.Ranking.Candidate n} → Fintype (GGSG19TopThree.TopWSelectionPair ρ W)

value:
fun {n : ℕ} {ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n}
    {W : AppliedModelingLib.SocialChoice.Ranking.Candidate n} =>
  id inferInstance
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.TopWSelectionPair

GGSG19TopThree.TopWSelectionPair.lo

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n} →
    {W : AppliedModelingLib.SocialChoice.Ranking.Candidate n} →
      GGSG19TopThree.TopWSelectionPair ρ W → AppliedModelingLib.SocialChoice.Ranking.Candidate n

value:
fun {n : ℕ} {ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n}
    {W : AppliedModelingLib.SocialChoice.Ranking.Candidate n} (pair : GGSG19TopThree.TopWSelectionPair ρ W) =>
  (↑pair).2
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.TopWSelectionPair

GGSG19TopThree.approvalPairwiseBase

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pUp pDown : ℝ) => 2 * √(pUp * pDown) + 1 - pUp - pDown

GGSG19TopThree.approvalPairwiseRate

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pUp pDown : ℝ) => AppliedModelingLib.Probability.ternaryGapClosedChernoffRate pUp pDown

Direct retained dependencies

- AppliedModelingLib.Probability.ternaryGapClosedChernoffRate

GGSG19TopThree.approvalPairwiseExtendedRate

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → WithTop ℝ

value:
fun (pUp pDown : ℝ) =>
  if GGSG19TopThree.approvalPairwiseBase pUp pDown = 0 then ⊤ else ↑(GGSG19TopThree.approvalPairwiseRate pUp pDown)

Direct retained dependencies

- GGSG19TopThree.approvalPairwiseBase
- GGSG19TopThree.approvalPairwiseRate

GGSG19TopThree.constructedWSelectionCandidateToRanking1

Declaration location: paper-local

Lean declaration body

type:
GGSG19TopThree.ConstructedWSelectionCandidate → GGSG19TopThree.ConstructedWSelectionRanking1Candidate

value:
fun (a : GGSG19TopThree.ConstructedWSelectionCandidate) =>
  match a with
  | GGSG19TopThree.ConstructedWSelectionCandidate.winner => 0
  | GGSG19TopThree.ConstructedWSelectionCandidate.loserI => 1
  | GGSG19TopThree.ConstructedWSelectionCandidate.loserJ => 2

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionRanking1Candidate

GGSG19TopThree.constructedWSelectionCrossTier

Declaration location: paper-local

Lean declaration body

type:
GGSG19TopThree.ConstructedWSelectionCandidate → GGSG19TopThree.ConstructedWSelectionCandidate → Prop

value:
```
fun (a a_1 : GGSG19TopThree.ConstructedWSelectionCandidate) =>
  match a, a_1 with
  | GGSG19TopThree.ConstructedWSelectionCandidate.winner, GGSG19TopThree.ConstructedWSelectionCandidate.loserI => True
  | GGSG19TopThree.ConstructedWSelectionCandidate.winner, GGSG19TopThree.ConstructedWSelectionCandidate.loserJ => True
  | x, x_1 => False
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionCandidate

GGSG19TopThree.constructedWSelectionPairLoserToRanking1

Declaration location: paper-local

Lean declaration body

type:
GGSG19TopThree.ConstructedWSelectionPair → GGSG19TopThree.ConstructedWSelectionRanking1Candidate

value:
```
fun (a : GGSG19TopThree.ConstructedWSelectionPair) =>
  match a with
  | GGSG19TopThree.ConstructedWSelectionPair.hi =>
    GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.loserI
  | GGSG19TopThree.ConstructedWSelectionPair.hj =>
    GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.loserJ
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.ConstructedWSelectionRanking1Candidate
- GGSG19TopThree.constructedWSelectionCandidateToRanking1

GGSG19TopThree.constructedWSelectionRankingMassReal

Declaration location: paper-local

Lean declaration body

type:
GGSG19TopThree.ConstructedWSelectionRanking → ℝ

value:
```
fun (a : GGSG19TopThree.ConstructedWSelectionRanking) =>
  match a with
  | GGSG19TopThree.ConstructedWSelectionRanking.hij => 46 / 1000
  | GGSG19TopThree.ConstructedWSelectionRanking.hji => 656 / 1000
  | GGSG19TopThree.ConstructedWSelectionRanking.ihj => 227 / 1000
  | GGSG19TopThree.ConstructedWSelectionRanking.ijh => 7 / 1000
  | GGSG19TopThree.ConstructedWSelectionRanking.jhi => 44 / 1000
  | GGSG19TopThree.ConstructedWSelectionRanking.jih => 20 / 1000
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionRanking

GGSG19TopThree.constructedWSelectionRankingMass

Declaration location: paper-local

Lean declaration body

type:
GGSG19TopThree.ConstructedWSelectionRanking → ENNReal

value:
```
fun (ranking : GGSG19TopThree.ConstructedWSelectionRanking) =>
  ENNReal.ofReal (GGSG19TopThree.constructedWSelectionRankingMassReal ranking)
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionRanking
- GGSG19TopThree.constructedWSelectionRankingMassReal

GGSG19TopThree.constructedWSelectionRankingLaw

Declaration location: paper-local

Lean declaration body

type:
PMF GGSG19TopThree.ConstructedWSelectionRanking

value:
PMF.ofFintype GGSG19TopThree.constructedWSelectionRankingMass GGSG19TopThree.constructedWSelectionRankingLaw._proof_1

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionRanking
- GGSG19TopThree.constructedWSelectionRankingMass

GGSG19TopThree.constructedWSelectionRankingToRanking1

Declaration location: paper-local

Lean declaration body

type:
GGSG19TopThree.ConstructedWSelectionRanking → GGSG19TopThree.ConstructedWSelectionRanking1

value:
```
fun (a : GGSG19TopThree.ConstructedWSelectionRanking) =>
  match a with
  | GGSG19TopThree.ConstructedWSelectionRanking.hij => Equiv.refl GGSG19TopThree.ConstructedWSelectionRanking1Candidate
  | GGSG19TopThree.ConstructedWSelectionRanking.hji => Equiv.swap 1 2
  | GGSG19TopThree.ConstructedWSelectionRanking.ihj => Equiv.swap 0 1
  | GGSG19TopThree.ConstructedWSelectionRanking.ijh => Equiv.trans (Equiv.swap 0 1) (Equiv.swap 0 2)
  | GGSG19TopThree.ConstructedWSelectionRanking.jhi => Equiv.trans (Equiv.swap 0 2) (Equiv.swap 0 1)
  | GGSG19TopThree.ConstructedWSelectionRanking.jih => Equiv.swap 0 2
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- GGSG19TopThree.ConstructedWSelectionRanking
- GGSG19TopThree.ConstructedWSelectionRanking1
- GGSG19TopThree.ConstructedWSelectionRanking1Candidate

GGSG19TopThree.constructedWSelectionRanking1Law

Declaration location: paper-local

Lean declaration body

type:
PMF GGSG19TopThree.ConstructedWSelectionRanking1

value:
PMF.map GGSG19TopThree.constructedWSelectionRankingToRanking1 GGSG19TopThree.constructedWSelectionRankingLaw

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionRanking
- GGSG19TopThree.ConstructedWSelectionRanking1
- GGSG19TopThree.constructedWSelectionRankingLaw
- GGSG19TopThree.constructedWSelectionRankingToRanking1

GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalDown

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.ConstructedWSelectionPair → ℝ

value:
fun (pair : GGSG19TopThree.ConstructedWSelectionPair) =>
  (AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb GGSG19TopThree.constructedWSelectionRanking1Law 1
        (GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.winner)
        (GGSG19TopThree.constructedWSelectionPairLoserToRanking1 pair) +
      AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb GGSG19TopThree.constructedWSelectionRanking1Law 2
        (GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.winner)
        (GGSG19TopThree.constructedWSelectionPairLoserToRanking1 pair)) /
    2
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb
- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.constructedWSelectionCandidateToRanking1
- GGSG19TopThree.constructedWSelectionPairLoserToRanking1
- GGSG19TopThree.constructedWSelectionRanking1Law

GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalUp

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.ConstructedWSelectionPair → ℝ

value:
fun (pair : GGSG19TopThree.ConstructedWSelectionPair) =>
  (AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb GGSG19TopThree.constructedWSelectionRanking1Law 1
        (GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.winner)
        (GGSG19TopThree.constructedWSelectionPairLoserToRanking1 pair) +
      AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb GGSG19TopThree.constructedWSelectionRanking1Law 2
        (GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.winner)
        (GGSG19TopThree.constructedWSelectionPairLoserToRanking1 pair)) /
    2
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb
- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.constructedWSelectionCandidateToRanking1
- GGSG19TopThree.constructedWSelectionPairLoserToRanking1
- GGSG19TopThree.constructedWSelectionRanking1Law

GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalRate

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.ConstructedWSelectionPair → ℝ

value:
fun (pair : GGSG19TopThree.ConstructedWSelectionPair) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalUp pair)
    (GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalDown pair)
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalDown
- GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalUp

GGSG19TopThree.constructedWSelectionRanking1StaticKDown

Declaration location: paper-local

Lean declaration body

```
type:
Fin 4 → GGSG19TopThree.ConstructedWSelectionPair → ℝ

value:
fun (K : Fin 4) (pair : GGSG19TopThree.ConstructedWSelectionPair) =>
  AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb GGSG19TopThree.constructedWSelectionRanking1Law (↑K)
    (GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.winner)
    (GGSG19TopThree.constructedWSelectionPairLoserToRanking1 pair)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb
- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.constructedWSelectionCandidateToRanking1
- GGSG19TopThree.constructedWSelectionPairLoserToRanking1
- GGSG19TopThree.constructedWSelectionRanking1Law

GGSG19TopThree.constructedWSelectionRanking1StaticKUp

Declaration location: paper-local

Lean declaration body

```
type:
Fin 4 → GGSG19TopThree.ConstructedWSelectionPair → ℝ

value:
fun (K : Fin 4) (pair : GGSG19TopThree.ConstructedWSelectionPair) =>
  AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb GGSG19TopThree.constructedWSelectionRanking1Law (↑K)
    (GGSG19TopThree.constructedWSelectionCandidateToRanking1 GGSG19TopThree.ConstructedWSelectionCandidate.winner)
    (GGSG19TopThree.constructedWSelectionPairLoserToRanking1 pair)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb
- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.constructedWSelectionCandidateToRanking1
- GGSG19TopThree.constructedWSelectionPairLoserToRanking1
- GGSG19TopThree.constructedWSelectionRanking1Law

GGSG19TopThree.constructedWSelectionRanking1StaticKRate

Declaration location: paper-local

Lean declaration body

```
type:
Fin 4 → GGSG19TopThree.ConstructedWSelectionPair → ℝ

value:
fun (K : Fin 4) (pair : GGSG19TopThree.ConstructedWSelectionPair) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.constructedWSelectionRanking1StaticKUp K pair)
    (GGSG19TopThree.constructedWSelectionRanking1StaticKDown K pair)
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.constructedWSelectionRanking1StaticKDown
- GGSG19TopThree.constructedWSelectionRanking1StaticKUp

GGSG19TopThree.durhamWard1Prob

Declaration location: paper-local

Lean declaration body

```
type:
ℕ → ℝ

value:
fun (n : ℕ) => ↑n / 1000
```

GGSG19TopThree.durhamWard1K3Down

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.DurhamWard1Pair → ℝ

value:
fun (a : GGSG19TopThree.DurhamWard1Pair) =>
  match a with
  | GGSG19TopThree.DurhamWard1Pair.hj => GGSG19TopThree.durhamWard1Prob 200
  | GGSG19TopThree.DurhamWard1Pair.ij => GGSG19TopThree.durhamWard1Prob 160
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1Prob

GGSG19TopThree.durhamWard1K3Up

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.DurhamWard1Pair → ℝ

value:
fun (a : GGSG19TopThree.DurhamWard1Pair) =>
  match a with
  | GGSG19TopThree.DurhamWard1Pair.hj => GGSG19TopThree.durhamWard1Prob 277
  | GGSG19TopThree.DurhamWard1Pair.ij => GGSG19TopThree.durhamWard1Prob 255
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1Prob

GGSG19TopThree.durhamWard1K3Rate

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.DurhamWard1Pair → ℝ

value:
fun (pair : GGSG19TopThree.DurhamWard1Pair) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.durhamWard1K3Up pair) (GGSG19TopThree.durhamWard1K3Down pair)
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.durhamWard1K3Down
- GGSG19TopThree.durhamWard1K3Up

GGSG19TopThree.durhamWard1K4Down

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.DurhamWard1Pair → ℝ

value:
fun (a : GGSG19TopThree.DurhamWard1Pair) =>
  match a with
  | GGSG19TopThree.DurhamWard1Pair.hj => GGSG19TopThree.durhamWard1Prob 188
  | GGSG19TopThree.DurhamWard1Pair.ij => GGSG19TopThree.durhamWard1Prob 217
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1Prob

GGSG19TopThree.durhamWard1K4Up

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.DurhamWard1Pair → ℝ

value:
fun (a : GGSG19TopThree.DurhamWard1Pair) =>
  match a with
  | GGSG19TopThree.DurhamWard1Pair.hj => GGSG19TopThree.durhamWard1Prob 266
  | GGSG19TopThree.DurhamWard1Pair.ij => GGSG19TopThree.durhamWard1Prob 295
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1Prob

GGSG19TopThree.durhamWard1K4Rate

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.DurhamWard1Pair → ℝ

value:
fun (pair : GGSG19TopThree.DurhamWard1Pair) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.durhamWard1K4Up pair) (GGSG19TopThree.durhamWard1K4Down pair)
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.durhamWard1K4Down
- GGSG19TopThree.durhamWard1K4Up

GGSG19TopThree.durhamWard1RandomizedDown

Declaration location: paper-local

Lean declaration body

type:
```
GGSG19TopThree.DurhamWard1Pair → ℝ
```

value:
```
fun (a : GGSG19TopThree.DurhamWard1Pair) =>
  match a with
  | GGSG19TopThree.DurhamWard1Pair.hj => GGSG19TopThree.durhamWard1Prob 194
  | GGSG19TopThree.DurhamWard1Pair.ij => GGSG19TopThree.durhamWard1Prob 189
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1Prob

GGSG19TopThree.durhamWard1RandomizedUp

Declaration location: paper-local

Lean declaration body

type:
```
GGSG19TopThree.DurhamWard1Pair → ℝ
```

value:
```
fun (a : GGSG19TopThree.DurhamWard1Pair) =>
  match a with
  | GGSG19TopThree.DurhamWard1Pair.hj => GGSG19TopThree.durhamWard1Prob 271
  | GGSG19TopThree.DurhamWard1Pair.ij => GGSG19TopThree.durhamWard1Prob 275
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1Prob

GGSG19TopThree.durhamWard1RandomizedRate

Declaration location: paper-local

Lean declaration body

type:
```
GGSG19TopThree.DurhamWard1Pair → ℝ
```

value:
```
fun (pair : GGSG19TopThree.DurhamWard1Pair) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.durhamWard1RandomizedUp pair)
    (GGSG19TopThree.durhamWard1RandomizedDown pair)
```

Direct retained dependencies

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.durhamWard1RandomizedDown
- GGSG19TopThree.durhamWard1RandomizedUp

GGSG19TopThree.finitePMFIidPathMeasure

Declaration location: paper-local

Lean declaration body

type:
```
{Signal : Type u_1} →
  [inst : MeasurableSpace Signal] →
    [Fintype Signal] → [DecidableEq Signal] → PMF Signal → MeasureTheory.Measure (ℕ → Signal)
```

value:
```
fun {Signal : Type u_1} [MeasurableSpace Signal] [Fintype Signal] [DecidableEq Signal] (law : PMF Signal) =>
  MeasureTheory.Measure.infinitePi fun (x : ℕ) => law.toMeasure
```

GGSG19TopThree.finiteUnivNonempty

Declaration location: paper-local

Lean declaration body

type:
∀ {ι : Type u_1} [inst : Fintype ι] [Nonempty ι], Finset.univ.Nonempty

value:
fun {ι : Type u_1} [Fintype ι] [Nonempty ι] =>
  Exists.intro (Classical.choice inferInstance) (of_eq_true (Finset.mem_univ._simp_1 (Classical.choice inferInstance)))

GGSG19TopThree.finiteOutcomeLearningExtendedRate

Declaration location: paper-local

Lean declaration body

type:
{Pair : Type u_1} → [Fintype Pair] → [Nonempty Pair] → (Pair → WithTop ℝ) → WithTop ℝ

value:
fun {Pair : Type u_1} [Fintype Pair] [Nonempty Pair] (rate : Pair → WithTop ℝ) =>
  Finset.univ.inf' GGSG19TopThree.finiteUnivNonempty rate

Direct retained dependencies

- GGSG19TopThree.finiteUnivNonempty

GGSG19TopThree.finiteOutcomeLearningRate

Declaration location: paper-local

Lean declaration body

type:
{Pair : Type u_1} → [Fintype Pair] → [Nonempty Pair] → (Pair → ℝ) → ℝ

value:
fun {Pair : Type u_1} [Fintype Pair] [Nonempty Pair] (rate : Pair → ℝ) =>
  Finset.univ.inf' GGSG19TopThree.finiteUnivNonempty rate

Direct retained dependencies

- GGSG19TopThree.finiteUnivNonempty

GGSG19TopThree.iidUniformTieCandidateScore

Declaration location: paper-local

Lean declaration body

type:
{Candidate : Type u_1} →
  {Signal : Type u_2} → {Tie : Type u_3} → (Candidate → Signal → ℝ) → (ℕ → Signal × Tie) → ℕ → Candidate → ℝ

value:
fun {Candidate : Type u_1} {Signal : Type u_2} {Tie : Type u_3} (score : Candidate → Signal → ℝ)
    (path : ℕ → Signal × Tie) (N : ℕ) (candidate : Candidate) =>
  ∑ voter ∈ Finset.range N, score candidate (path voter).1

GGSG19TopThree.mallowsBumpPosition

Declaration location: paper-local

Lean declaration body

type:
{N : ℕ} → Fin N → Fin N → Fin N

value:
fun {N : ℕ} (inserted current : Fin N) =>
  if ↑inserted ≤ ↑current then if hnext : ↑current + 1 < N then ⟨↑current + 1, hnext⟩ else current else current

GGSG19TopThree.mallowsHighNoisePhi

Declaration location: paper-local

Lean declaration body

type:
ℝ

value:
4 / 5

GGSG19TopThree.mallowsInsertionNormalizer

Declaration location: paper-local

Lean declaration body

type:
{N : ℕ} → ℝ → Fin N → ℝ

value:
fun {N : ℕ} (q : ℝ) (stage : Fin N) => ∑ r ∈ Finset.range (↑stage + 1), q ^ (↑stage - r)

GGSG19TopThree.mallowsInsertionWeight

Declaration location: paper-local

Lean declaration body

type:
{N : ℕ} → ℝ → Fin N → Fin N → ℝ

value:
fun {N : ℕ} (q : ℝ) (stage inserted : Fin N) => if ↑inserted ≤ ↑stage then q ^ (↑stage - ↑inserted) else 0

GGSG19TopThree.mallowsInsertionPMF

Declaration location: paper-local

Lean declaration body

type:
{N : ℕ} → (q : ℝ) → 0 ≤ q → Fin N → PMF (Fin N)

value:
fun {N : ℕ} (q : ℝ) (hq : 0 ≤ q) (stage : Fin N) =>
  PMF.ofFintype
    (fun (inserted : Fin N) =>
      ENNReal.ofReal
        (GGSG19TopThree.mallowsInsertionWeight q stage inserted / GGSG19TopThree.mallowsInsertionNormalizer q stage))
    (GGSG19TopThree.mallowsInsertionPMF._proof_1 q hq stage)

Direct retained dependencies

- GGSG19TopThree.mallowsInsertionNormalizer
- GGSG19TopThree.mallowsInsertionWeight

GGSG19TopThree.mallowsPairInitialState

Declaration location: paper-local

Lean declaration body

type:
(N : ℕ) → GGSG19TopThree.MallowsPairPositionState N

value:
fun (N : ℕ) => (none, none)

Direct retained dependencies

- GGSG19TopThree.MallowsPairPositionState

GGSG19TopThree.mallowsTopThreePairUpProb

Declaration location: paper-local

Lean declaration body

```
type:
AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2 →
  AppliedModelingLib.SocialChoice.Ranking.Candidate 2 → AppliedModelingLib.SocialChoice.Ranking.Candidate 2 → ℝ

value:
fun (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2)
    (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate 2) =>
  AppliedModelingLib.pmfProb M.law fun (π : AppliedModelingLib.SocialChoice.Ranking.Ranking 2) =>
    AppliedModelingLib.SocialChoice.Ranking.rankOf π hi < 3 ∧ AppliedModelingLib.SocialChoice.Ranking.rankOf π lo = 3
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- AppliedModelingLib.pmfProb

GGSG19TopThree.mallowsTopTwoPairUpProb

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n →
    AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  ∑ other : AppliedModelingLib.SocialChoice.Ranking.Candidate n,
    if other ≠ hi ∧ other ≠ lo then M.firstSecondChoiceProb hi other + M.firstSecondChoiceProb other hi else 0
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.firstSecondChoiceProb

GGSG19TopThree.mallowsTopWKApprovalPairDownProb

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n) →
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) → ℕ → GGSG19TopThree.TopWSelectionPair M.center W → ℝ

value:
fun {n : ℕ} (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (K : ℕ)
    (pair : GGSG19TopThree.TopWSelectionPair M.center W) =>
  AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb M.law K pair.hi pair.lo
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.TopWSelectionPair.hi
- GGSG19TopThree.TopWSelectionPair.lo

GGSG19TopThree.mallowsTopWKApprovalPairUpProb

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n) →
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) → ℕ → GGSG19TopThree.TopWSelectionPair M.center W → ℝ

value:
fun {n : ℕ} (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (K : ℕ)
    (pair : GGSG19TopThree.TopWSelectionPair M.center W) =>
  AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb M.law K pair.hi pair.lo
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.TopWSelectionPair.hi
- GGSG19TopThree.TopWSelectionPair.lo

GGSG19TopThree.mallowsTopWKApprovalPairRate

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n) →
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) → ℕ → GGSG19TopThree.TopWSelectionPair M.center W → ℝ

value:
fun {n : ℕ} (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (K : ℕ)
    (pair : GGSG19TopThree.TopWSelectionPair M.center W) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.mallowsTopWKApprovalPairUpProb M W K pair)
    (GGSG19TopThree.mallowsTopWKApprovalPairDownProb M W K pair)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.mallowsTopWKApprovalPairDownProb
- GGSG19TopThree.mallowsTopWKApprovalPairUpProb

GGSG19TopThree.mallowsTopWKApprovalOutcomeRate

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n →
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) → 0 < ↑W → ℕ → ℝ

value:
fun {n : ℕ} (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (hW : 0 < ↑W) (K : ℕ) =>
  GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.mallowsTopWKApprovalPairRate M W K)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.TopWSelectionPair.instFintype
- GGSG19TopThree.finiteOutcomeLearningRate
- GGSG19TopThree.mallowsTopWKApprovalPairRate

GGSG19TopThree.mallowsTopWRandomizedKApprovalPairDownProb

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {Rule : Type u_1} →
    [Fintype Rule] →
      (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n) →
        (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) →
          (Rule → ℕ) → (Rule → ℝ) → GGSG19TopThree.TopWSelectionPair M.center W → ℝ

value:
fun {n : ℕ} {Rule : Type u_1} [Fintype Rule] (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (K : Rule → ℕ) (weight : Rule → ℝ)
    (pair : GGSG19TopThree.TopWSelectionPair M.center W) =>
  ∑ rule : Rule, weight rule * GGSG19TopThree.mallowsTopWKApprovalPairDownProb M W (K rule) pair
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.mallowsTopWKApprovalPairDownProb

GGSG19TopThree.mallowsTopWRandomizedKApprovalPairUpProb

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {Rule : Type u_1} →
    [Fintype Rule] →
      (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n) →
        (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) →
          (Rule → ℕ) → (Rule → ℝ) → GGSG19TopThree.TopWSelectionPair M.center W → ℝ

value:
fun {n : ℕ} {Rule : Type u_1} [Fintype Rule] (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (K : Rule → ℕ) (weight : Rule → ℝ)
    (pair : GGSG19TopThree.TopWSelectionPair M.center W) =>
  ∑ rule : Rule, weight rule * GGSG19TopThree.mallowsTopWKApprovalPairUpProb M W (K rule) pair
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.mallowsTopWKApprovalPairUpProb

GGSG19TopThree.mallowsTopWRandomizedKApprovalPairRate

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {Rule : Type u_1} →
    [Fintype Rule] →
      (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n) →
        (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) →
          (Rule → ℕ) → (Rule → ℝ) → GGSG19TopThree.TopWSelectionPair M.center W → ℝ
```

```
value:
fun {n : ℕ} {Rule : Type u_1} [Fintype Rule] (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (K : Rule → ℕ) (weight : Rule → ℝ)
    (pair : GGSG19TopThree.TopWSelectionPair M.center W) =>
  GGSG19TopThree.approvalPairwiseRate (GGSG19TopThree.mallowsTopWRandomizedKApprovalPairUpProb M W K weight pair)
    (GGSG19TopThree.mallowsTopWRandomizedKApprovalPairDownProb M W K weight pair)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.mallowsTopWRandomizedKApprovalPairDownProb
- GGSG19TopThree.mallowsTopWRandomizedKApprovalPairUpProb

GGSG19TopThree.mallowsTopWRandomizedKApprovalOutcomeRate

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  {Rule : Type u_1} →
    [Fintype Rule] →
      AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n →
        (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) → 0 < ↑W → (Rule → ℕ) → (Rule → ℝ) → ℝ
```

```
value:
fun {n : ℕ} {Rule : Type u_1} [Fintype Rule] (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (hW : 0 < ↑W) (K : Rule → ℕ) (weight : Rule → ℝ) =>
  GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.mallowsTopWRandomizedKApprovalPairRate M W K weight)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.TopWSelectionPair.instFintype
- GGSG19TopThree.finiteOutcomeLearningRate
- GGSG19TopThree.mallowsTopWRandomizedKApprovalPairRate

GGSG19TopThree.mallowsUpdateTrackedPosition

Declaration location: paper-local

Lean declaration body

```
type:
{N : ℕ} → Fin N → Fin N → Fin N → Option (Fin N) → Option (Fin N)
```

```
value:
fun {N : ℕ} (stage inserted trackedStage : Fin N) (current : Option (Fin N)) =>
  if stage = trackedStage then some inserted else Option.map (GGSG19TopThree.mallowsBumpPosition inserted) current
```

Direct retained dependencies

- GGSG19TopThree.mallowsBumpPosition

GGSG19TopThree.mallowsUpdatePairState

Declaration location: paper-local

Lean declaration body

```
type:
{N : ℕ} →
  Fin N → Fin N → Fin N → Fin N → GGSG19TopThree.MallowsPairPositionState N → GGSG19TopThree.MallowsPairPositionState N

value:
fun {N : ℕ} (leftStage rightStage stage inserted : Fin N) (state : GGSG19TopThree.MallowsPairPositionState N) =>
  (GGSG19TopThree.mallowsUpdateTrackedPosition stage inserted leftStage state.1,
    GGSG19TopThree.mallowsUpdateTrackedPosition stage inserted rightStage state.2)
```

Direct retained dependencies

- GGSG19TopThree.MallowsPairPositionState
- GGSG19TopThree.mallowsUpdateTrackedPosition

GGSG19TopThree.ProofBridge.mallowsFourCandidate34JointLocationProb

Declaration location: paper-local

Lean declaration body

```
type:
(q : ℝ) → 0 ≤ q → Fin 4 → Fin 4 → ℝ

value:
fun (q : ℝ) (hq : 0 ≤ q) (ell k : Fin 4) =>
  ∑ inserted3 : Fin 4,
    ∑ inserted4 : Fin 4,
      if
          GGSG19TopThree.mallowsUpdatePairState 2 3 3 inserted4
            (GGSG19TopThree.mallowsUpdatePairState 2 3 2 inserted3 (GGSG19TopThree.mallowsPairInitialState 4)) =
          (some ell, some k) then
        ((GGSG19TopThree.mallowsInsertionPMF q hq 2) inserted3).toReal *
          ((GGSG19TopThree.mallowsInsertionPMF q hq 3) inserted4).toReal
      else 0
```

Direct retained dependencies

- GGSG19TopThree.MallowsPairPositionState
- GGSG19TopThree.mallowsInsertionPMF
- GGSG19TopThree.mallowsPairInitialState
- GGSG19TopThree.mallowsUpdatePairState

GGSG19TopThree.mallowsPairExplicitTransitionMass

Declaration location: paper-local

Lean declaration body

```
type:
{N : ℕ} →
  GGSG19TopThree.MallowsRepeatedInsertionModel N →
    Fin N → Fin N → Fin N → GGSG19TopThree.MallowsPairPositionState N → GGSG19TopThree.MallowsPairPositionState N → ℝ

value:
fun {N : ℕ} (model : GGSG19TopThree.MallowsRepeatedInsertionModel N) (leftStage rightStage stage : Fin N)
    (previous target : GGSG19TopThree.MallowsPairPositionState N) =>
  ∑ inserted : Fin N,
    if target = GGSG19TopThree.mallowsUpdatePairState leftStage rightStage stage inserted previous then
      ((model.insertion stage) inserted).toReal
    else 0
```

Direct retained dependencies

- GGSG19TopThree.MallowsPairPositionState
- GGSG19TopThree.MallowsRepeatedInsertionModel
- GGSG19TopThree.mallowsUpdatePairState

GGSG19TopThree.mallowsPairExplicitTableStep

Declaration location: paper-local

Lean declaration body

```
type:
{N : ℕ} →
  GGSG19TopThree.MallowsRepeatedInsertionModel N →
    Fin N →
      Fin N → Fin N → (GGSG19TopThree.MallowsPairPositionState N → ℝ) → GGSG19TopThree.MallowsPairPositionState N → ℝ

value:
fun {N : ℕ} (model : GGSG19TopThree.MallowsRepeatedInsertionModel N) (leftStage rightStage stage : Fin N)
    (previousTable : GGSG19TopThree.MallowsPairPositionState N → ℝ) (a : GGSG19TopThree.MallowsPairPositionState N) =>
  ∑ previous : GGSG19TopThree.MallowsPairPositionState N,
    previousTable previous *
      GGSG19TopThree.mallowsPairExplicitTransitionMass model leftStage rightStage stage previous a
```

Direct retained dependencies

- GGSG19TopThree.MallowsPairPositionState
- GGSG19TopThree.MallowsRepeatedInsertionModel
- GGSG19TopThree.mallowsPairExplicitTransitionMass

GGSG19TopThree.mallowsW3LoserCandidate

Declaration location: paper-local

Lean declaration body

```
type:
AppliedModelingLib.SocialChoice.Ranking.Candidate 2

value:
3
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate

GGSG19TopThree.mallowsW3WinnerCandidate

Declaration location: paper-local

Lean declaration body

```
type:
Fin 3 → AppliedModelingLib.SocialChoice.Ranking.Candidate 2

value:
fun (i : Fin 3) => ⟨↑i, GGSG19TopThree.mallowsW3WinnerCandidate._proof_2 i⟩
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate

GGSG19TopThree.mallowsW3K1PairRate

Declaration location: paper-local

Lean declaration body

```
type:
AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2 → Fin 3 → ℝ

value:
fun (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2) (i : Fin 3) =>
  GGSG19TopThree.approvalPairwiseRate
    (AppliedModelingLib.SocialChoice.Ranking.firstChoiceProb M.law (GGSG19TopThree.mallowsW3WinnerCandidate i))
    (AppliedModelingLib.SocialChoice.Ranking.firstChoiceProb M.law GGSG19TopThree.mallowsW3LoserCandidate)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.firstChoiceProb
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.mallowsW3LoserCandidate
- GGSG19TopThree.mallowsW3WinnerCandidate

GGSG19TopThree.mallowsW3K2PairRate

Declaration location: paper-local

Lean declaration body

```
type:
AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2 → Fin 3 → ℝ

value:
fun (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2) (i : Fin 3) =>
  GGSG19TopThree.approvalPairwiseRate
    (GGSG19TopThree.mallowsTopTwoPairUpProb M (GGSG19TopThree.mallowsW3WinnerCandidate i)
      GGSG19TopThree.mallowsW3LoserCandidate)
    (GGSG19TopThree.mallowsTopTwoPairUpProb M GGSG19TopThree.mallowsW3LoserCandidate
      (GGSG19TopThree.mallowsW3WinnerCandidate i))
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.mallowsTopTwoPairUpProb
- GGSG19TopThree.mallowsW3LoserCandidate
- GGSG19TopThree.mallowsW3WinnerCandidate

GGSG19TopThree.mallowsW3K3PairRate

Declaration location: paper-local

Lean declaration body

```
type:
AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2 → Fin 3 → ℝ

value:
fun (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2) (i : Fin 3) =>
  GGSG19TopThree.approvalPairwiseRate
    (GGSG19TopThree.mallowsTopThreePairUpProb M (GGSG19TopThree.mallowsW3WinnerCandidate i)
      GGSG19TopThree.mallowsW3LoserCandidate)
    (GGSG19TopThree.mallowsTopThreePairUpProb M GGSG19TopThree.mallowsW3LoserCandidate
      (GGSG19TopThree.mallowsW3WinnerCandidate i))
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.mallowsTopThreePairUpProb
- GGSG19TopThree.mallowsW3LoserCandidate
- GGSG19TopThree.mallowsW3WinnerCandidate

GGSG19TopThree.mallowsW3StaticKApprovalPairRate

Declaration location: paper-local

Lean declaration body

type:
AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2 → Fin 3 → Fin 3 → ℝ

value:
fun (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2) (rule i : Fin 3) =>
  if rule = 0 then GGSG19TopThree.mallowsW3K1PairRate M i
  else if rule = 1 then GGSG19TopThree.mallowsW3K2PairRate M i else GGSG19TopThree.mallowsW3K3PairRate M i

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- GGSG19TopThree.mallowsW3K1PairRate
- GGSG19TopThree.mallowsW3K2PairRate
- GGSG19TopThree.mallowsW3K3PairRate

GGSG19TopThree.pairwiseScoringErrorProb

Declaration location: paper-local

Lean declaration body

type:
{Signal : Type u_1} → [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Signal → ℝ) → (Signal → ℝ) → ℕ → ℝ

value:
fun {Signal : Type u_1} [Fintype Signal] [DecidableEq Signal] (law : PMF Signal) (hiScore loScore : Signal → ℝ)
    (n : ℕ) =>
  AppliedModelingLib.Probability.finitelidScoreGapLeftTailProb law hiScore loScore n

Direct retained dependencies

- AppliedModelingLib.Probability.finiteIidScoreGapLeftTailProb

GGSG19TopThree.finiteScoreGapPairwiseErrorProb

Declaration location: paper-local

Lean declaration body

type:
{Candidate : Type u_1} →
  {Signal : Type u_2} →
    [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Candidate → Signal → ℝ) → Candidate → Candidate → ℕ → ℝ

value:
fun {Candidate : Type u_1} {Signal : Type u_2} [Fintype Signal] [DecidableEq Signal] (law : PMF Signal)
    (score : Candidate → Signal → ℝ) (hi lo : Candidate) (n : ℕ) =>
  GGSG19TopThree.pairwiseScoringErrorProb law (score hi) (score lo) n

Direct retained dependencies

- GGSG19TopThree.pairwiseScoringErrorProb

GGSG19TopThree.ProofBridge.zeroNoiseTopWApprovalError

Declaration location: paper-local

Lean declaration body

type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℕ → ℝ

value:
fun {n : ℕ} (center : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (sampleSize : ℕ) =>
  ∑ pair : GGSG19TopThree.TopWSelectionPair center W,
    GGSG19TopThree.finiteScoreGapPairwiseErrorProb (PMF.pure center)
      (fun (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n)
          (ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
        AppliedModelingLib.SocialChoice.Ranking.kApprovalScore (↑W) ranking candidate)
      pair.hi pair.lo sampleSize

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.kApprovalScore
- GGSG19TopThree.TopWSelectionPair
- GGSG19TopThree.TopWSelectionPair.hi
- GGSG19TopThree.TopWSelectionPair.instFintype
- GGSG19TopThree.TopWSelectionPair.lo
- GGSG19TopThree.finiteScoreGapPairwiseErrorProb

GGSG19TopThree.pairwiseScoringExtendedRate

Declaration location: paper-local

Lean declaration body

type:
```
{Signal : Type u_1} → [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Signal → ℝ) → (Signal → ℝ) → WithTop ℝ
```

value:
```
fun {Signal : Type u_1} [Fintype Signal] [DecidableEq Signal] (law : PMF Signal) (hiScore loScore : Signal → ℝ) =>
  AppliedModelingLib.Probability.finiteRateFunctionTop law (fun (signal : Signal) => hiScore signal - loScore signal) 0
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteRateFunctionTop

GGSG19TopThree.pairwiseScoringRate

Declaration location: paper-local

Lean declaration body

type:
```
{Signal : Type u_1} → [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Signal → ℝ) → (Signal → ℝ) → ℝ
```

value:
```
fun {Signal : Type u_1} [Fintype Signal] [DecidableEq Signal] (law : PMF Signal) (hiScore loScore : Signal → ℝ) =>
  AppliedModelingLib.Probability.finiteScoreGapChernoffRate law hiScore loScore
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteScoreGapChernoffRate

GGSG19TopThree.prefixProbFromEvent

Declaration location: paper-local

Lean declaration body

type:
```
{Cut : Type u_1} →
  {Candidate : Type u_2} →
    {Signal : Type u_3} →
      [Fintype Signal] →
        [DecidableEq Signal] →
          PMF Signal →
            (inPrefix : Signal → Candidate → Cut → Prop) →
              [(signal : Signal) → (candidate : Candidate) → (cut : Cut) → Decidable (inPrefix signal candidate cut)] →
                Candidate → Cut → ℝ
```

value:
```
fun {Cut : Type u_1} {Candidate : Type u_2} {Signal : Type u_3} [Fintype Signal] [DecidableEq Signal] (law : PMF Signal)
    (inPrefix : Signal → Candidate → Cut → Prop)
    [(signal : Signal) → (candidate : Candidate) → (cut : Cut) → Decidable (inPrefix signal candidate cut)]
    (candidate : Candidate) (cut : Cut) =>
  AppliedModelingLib.pmfProb law fun (signal : Signal) => inPrefix signal candidate cut
```

Direct retained dependencies

- AppliedModelingLib.pmfProb

GGSG19TopThree.prefixScoreFromEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Cut : Type u_1} →
  {Candidate : Type u_2} →
    {Signal : Type u_3} →
      [Fintype Cut] →
        (Cut → ℝ) →
          (inPrefix : Signal → Candidate → Cut → Prop) →
            [(signal : Signal) → (candidate : Candidate) → (cut : Cut) → Decidable (inPrefix signal candidate cut)] →
              Candidate → Signal → ℝ

value:
fun {Cut : Type u_1} {Candidate : Type u_2} {Signal : Type u_3} [Fintype Cut] (diff : Cut → ℝ)
    (inPrefix : Signal → Candidate → Cut → Prop)
    [(signal : Signal) → (candidate : Candidate) → (cut : Cut) → Decidable (inPrefix signal candidate cut)]
    (candidate : Candidate) (signal : Signal) =>
  ∑ cut : Cut, diff cut * if inPrefix signal candidate cut then 1 else 0
```

GGSG19TopThree.randomizedScoringMixtureMGF

Declaration location: paper-local

Lean declaration body

```
type:
{Rule : Type u_1} →
  {Signal : Type u_2} →
    [Fintype Rule] → [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Rule → ℝ) → (Rule → Signal → ℝ) → ℝ → ℝ

value:
fun {Rule : Type u_1} {Signal : Type u_2} [Fintype Rule] [Fintype Signal] [DecidableEq Signal] (law : PMF Signal)
    (weight : Rule → ℝ) (gap : Rule → Signal → ℝ) (z : ℝ) =>
  ∑ rule : Rule, weight rule * AppliedModelingLib.Probability.finiteMGF law (gap rule) z
```

Direct retained dependencies

- AppliedModelingLib.Probability.finiteMGF

GGSG19TopThree.randomizedScoringMixtureLogMGF

Declaration location: paper-local

Lean declaration body

```
type:
{Rule : Type u_1} →
  {Signal : Type u_2} →
    [Fintype Rule] → [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Rule → ℝ) → (Rule → Signal → ℝ) → ℝ → ℝ

value:
fun {Rule : Type u_1} {Signal : Type u_2} [Fintype Rule] [Fintype Signal] [DecidableEq Signal] (law : PMF Signal)
    (weight : Rule → ℝ) (gap : Rule → Signal → ℝ) (z : ℝ) =>
  Real.log (GGSG19TopThree.randomizedScoringMixtureMGF law weight gap z)
```

Direct retained dependencies

- GGSG19TopThree.randomizedScoringMixtureMGF

GGSG19TopThree.randomizedScoringMixtureRate

Declaration location: paper-local

Lean declaration body

```
type:
{Rule : Type u_1} →
  {Signal : Type u_2} →
    [Fintype Rule] → [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Rule → ℝ) → (Rule → Signal → ℝ) → ℝ

value:
fun {Rule : Type u_1} {Signal : Type u_2} [Fintype Rule] [Fintype Signal] [DecidableEq Signal] (law : PMF Signal)
    (weight : Rule → ℝ) (gap : Rule → Signal → ℝ) =>
  -sInf (Set.range fun (z : ℝ) => GGSG19TopThree.randomizedScoringMixtureLogMGF law weight gap z)
```

Direct retained dependencies

- GGSG19TopThree.randomizedScoringMixtureLogMGF

GGSG19TopThree.randomizedScoringSamplingLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Rule : Type u_1} →
  {Signal : Type u_2} →
    [inst : Fintype Rule] →
      [DecidableEq Rule] →
        [Fintype Signal] →
          [DecidableEq Signal] →
            PMF Signal →
              (weight : Rule → ℝ) →
                (∀ (rule : Rule), 0 ≤ weight rule) → ∑ rule : Rule, weight rule = 1 → PMF (Rule × Signal)

value:
fun {Rule : Type u_1} {Signal : Type u_2} [Fintype Rule] [DecidableEq Rule] [Fintype Signal] [DecidableEq Signal]
    (law : PMF Signal) (weight : Rule → ℝ) (hweight : ∀ (rule : Rule), 0 ≤ weight rule)
    (hsum : ∑ rule : Rule, weight rule = 1) =>
  have ruleLaw : PMF Rule :=
    AppliedModelingLib.finiteWeightedPMF weight hweight
      (GGSG19TopThree.randomizedScoringSamplingLaw._proof_1 weight hsum);
  ruleLaw.bind fun (rule : Rule) => PMF.map (fun (signal : Signal) => (rule, signal)) law
```

Direct retained dependencies

- AppliedModelingLib.finiteWeightedPMF

GGSG19TopThree.rankingInTopPrefix

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  AppliedModelingLib.SocialChoice.Ranking.Ranking n →
    AppliedModelingLib.SocialChoice.Ranking.Candidate n → GGSG19TopThree.RankingProperPrefixCut n → Prop

value:
fun {n : ℕ} (ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (cut : GGSG19TopThree.RankingProperPrefixCut n) =>
  ↑(AppliedModelingLib.SocialChoice.Ranking.rankOf ranking candidate) ≤ ↑cut
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.RankingProperPrefixCut

GGSG19TopThree.rankingInTopPrefix_decidable

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  (ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n) →
      (cut : GGSG19TopThree.RankingProperPrefixCut n) →
        Decidable (GGSG19TopThree.rankingInTopPrefix ranking candidate cut)

value:
fun {n : ℕ} (ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (cut : GGSG19TopThree.RankingProperPrefixCut n) =>
  id inferInstance
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.RankingProperPrefixCut
- GGSG19TopThree.rankingInTopPrefix

GGSG19TopThree.rankingTopPrefixProb

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    AppliedModelingLib.SocialChoice.Ranking.Candidate n → GGSG19TopThree.RankingProperPrefixCut n → ℝ

value:
fun {n : ℕ} (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n))
    (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (cut : GGSG19TopThree.RankingProperPrefixCut n) =>
  GGSG19TopThree.prefixProbFromEvent law GGSG19TopThree.rankingInTopPrefix candidate cut
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.RankingProperPrefixCut
- GGSG19TopThree.prefixProbFromEvent
- GGSG19TopThree.rankingInTopPrefix
- GGSG19TopThree.rankingInTopPrefix_decidable

GGSG19TopThree.constructedWSelectionRanking1TopPrefixProb

Declaration location: paper-local

Lean declaration body

```
type:
GGSG19TopThree.ConstructedWSelectionCandidate → Fin 2 → ℝ

value:
fun (a : GGSG19TopThree.ConstructedWSelectionCandidate) (a_1 : Fin 2) =>
  GGSG19TopThree.rankingTopPrefixProb GGSG19TopThree.constructedWSelectionRanking1Law
    (GGSG19TopThree.constructedWSelectionCandidateToRanking1 a) a_1
```

Direct retained dependencies

- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.constructedWSelectionCandidateToRanking1
- GGSG19TopThree.constructedWSelectionRanking1Law
- GGSG19TopThree.rankingTopPrefixProb

GGSG19TopThree.PropositionOneUniformTieConsistency

Declaration location: paper-local

Lean declaration body

```
field allReasonableRules:
∀ {n Stage : ℕ} {law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)}
  {targetPrefix : Fin Stage → Finset (AppliedModelingLib.SocialChoice.Ranking.Candidate n)},
  GGSG19TopThree.PropositionOneUniformTieConsistency law targetPrefix →
    ∀ (diff : GGSG19TopThree.RankingProperPrefixCut n → ℝ),
      GGSG19TopThree.ReasonablePrefixWeights diff →
        ∀ᵐ (path :
          ℕ →
            AppliedModelingLib.SocialChoice.Ranking.Ranking n ×
              AppliedModelingLib.SocialChoice.Ranking.Ranking
                n) ∂GGSG19TopThree.finitePMFIidPathMeasure (GGSG19TopThree.rankingUniformTieSignalLaw law),
          ∀ᶠ (N : ℕ) in Filter.atTop,
            GGSG19TopThree.iidUniformTieTieredCorrect (GGSG19TopThree.rankingPrefixScore diff) targetPrefix path N
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.RankingProperPrefixCut
- GGSG19TopThree.ReasonablePrefixWeights
- GGSG19TopThree.finitePMFIidPathMeasure
- GGSG19TopThree.iidUniformTieTieredCorrect
- GGSG19TopThree.rankingPrefixScore
- GGSG19TopThree.rankingUniformTieSignalLaw

GGSG19TopThree.topWBoundaryPair

Declaration location: paper-local

Lean declaration body

```
type:
{n : ℕ} →
  (ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) → 0 < ↑W → GGSG19TopThree.TopWSelectionPair ρ W

value:
fun {n : ℕ} (ρ : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) (hW : 0 < ↑W) =>
  let pred : AppliedModelingLib.SocialChoice.Ranking.Candidate n :=
    ⟨↑W - 1, GGSG19TopThree.topWBoundaryPair._proof_2 W hW⟩;
  ⟨(ρ pred, ρ W), GGSG19TopThree.topWBoundaryPair._proof_3 ρ W hW⟩
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.TopWSelectionPair

Material library prerequisites

AppliedModelingLib.SocialChoice.Ranking.kApprovalScore

Source locator: cited publication, lines 330-339

Verbatim paper-source connection

```
\item [$K$-Approval] Voter $v$ selects her favorite $K$ candidates, i.e., reveals $\{i \,:\, {\sigma_v}(i) \leq K \}$. A candidate receives a score $s_{iv} =
↪  1$ for being selected, $0$ otherwise.
%

%
    %


\end{description}

$\beta$ encodes both elicitation and aggregation. For example, $K$-Approval is equivalent to $K$-ranking with score function $\beta(k) = \bbI[k \leq K]$.
↪  Furthermore, note that given $K$-ranking data, one can simulate $K'$-ranking elicitation for $K'\leq K$ with a $\beta$ s.t. $\beta(k) = 0$ for $k>K'$. %
```

Lean-expanded library semantic target

```
type:
{n : ℕ} →
  ℕ → AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (K : ℕ) (π : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (c : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  if AppliedModelingLib.SocialChoice.Ranking.approvedByK K π c then 1 else 0
```

Exact Lean library declaration

```
/-- Real-valued K-approval score indicator. -/
def kApprovalScore {n : ℕ} (K : ℕ) (π : Ranking n) (c : Candidate n) : ℝ :=
  if approvedByK K π c then 1 else 0
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.approvedByK
- AppliedModelingLib.SocialChoice.Ranking.instDecidablePredApprovedByK

Recorded source-to-library screening **Verdict:** matches

Reason: The K-Approval definition at audit/cited publication:330-339 gives score 1 to the top K candidates and 0 to all others. The actual library function returns precisely those two values according to approvedByK. That predicate expands to rankOf pi c < K; rankOf is the inverse permutation and ranks are zero-based, so this is equivalent to source sigma_v(i)<=K. The displayed paper use sets K to the selected winner count W and imposes W>0; the score and cutoff therefore retain exactly the source meaning in its use as well as for general K.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Paper-specific semantic prerequisites

GGSG19TopThree.ProofBridge.iidUniformTieCandidateAverageScore

Paper-local declaration:
GGSG19TopThree.ProofBridge.iidUniformTieCandidateAverageScore
Source locator: cited publication, lines 345-345

Verbatim paper-source connection

```
\parbold{Outcome} After $N$ voters, candidate $i$'s cumulative score is $s_i^N = \frac{1}{N} \sum_{v=1}^N s_{iv}$. Candidates are ranked in
↪  descending order of score, to form ranking ${\sigma}^N$, with ties broken uniformly at random. We denote the \textit{outcome} after $N$ voters,
↪  corresponding to the goal $G$, as $O^N(M, F, \beta, G)$. For example, for the goal of selecting $W$ winners, $O^N(M, F, \beta, G)$ is simply the top
↪  $W$ candidates in ${\sigma}^N$. When $(M, F, \beta, G)$ is clear from context, we will refer to the outcome as $O^N$.
```

Lean-expanded paper semantic target

```
type:
{Candidate : Type u_1} →
  {Signal : Type u_2} → {Tie : Type u_3} → (Candidate → Signal → ℝ) → (ℕ → Signal × Tie) → ℕ → Candidate → ℝ

value:
fun {Candidate : Type u_1} {Signal : Type u_2} {Tie : Type u_3} (score : Candidate → Signal → ℝ)
    (path : ℕ → Signal × Tie) (N : ℕ) (candidate : Candidate) =>
  (↑N)⁻¹ * GGSG19TopThree.iidUniformTieCandidateScore score path N candidate
```

Direct retained dependencies

- GGSG19TopThree.iidUniformTieCandidateScore

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source outcome paragraph at audit/cited publication:345 defines s_i^N as 1/N times the sum of the first N voter scores. The actual target multiplies the displayed iidUniformTieCandidateScore by the real reciprocal of N; that dependency sums score candidate (path voter).1 over Finset.range N. Thus the voter coordinate, number of terms, candidate argument, and normalization agree after zero-based indexing. The extra tie coordinate is ignored in score accumulation, as required.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.ProofBridge.kRankingScore

Paper-local declaration:
GGSG19TopThree.ProofBridge.kRankingScore
Source locator: cited publication, lines 325-329

Verbatim paper-source connection

```
\parbold{Elicitation and Aggregation}
Voters vote using an elicitation mechanism. Their votes are then aggregated using a positional scoring rule, parameterized as $\beta: \{1,\dots,M\}
↪  \mapsto \bbR$. We consider the following mechanisms:
\begin{description}
    \item [$K$-Ranking] Voter $v$ ranks her favorite $K$ candidates, i.e., reveals $\{(i, {\sigma_v}(i)) \,:\, {\sigma_v}(i) \leq K \}$. Candidate $i$ then
    ↪  receives a score $s_{iv} = \beta({\sigma_v}(i))$ if ranked, $0$ otherwise. For example, $\beta(k) = M-k$ for the Borda count.\footnote{In Borda,
    ↪  candidates not ranked receive a score $(M-K-1)/2$, consistent with assuming they are all tied in position $(K + 1)$.}
```

Lean-expanded paper semantic target

```
type:
{n : ℕ} →
  ℕ →
    (AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ) →
      AppliedModelingLib.SocialChoice.Ranking.Ranking n → AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ

value:
fun {n : ℕ} (K : ℕ) (beta : AppliedModelingLib.SocialChoice.Ranking.Candidate n → ℝ)
    (ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n) =>
  if ↑(AppliedModelingLib.SocialChoice.Ranking.rankOf ranking candidate) < K then
    beta (AppliedModelingLib.SocialChoice.Ranking.rankOf ranking candidate)
  else 0
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf

Recorded source-to-declaration screening **Verdict:** matches

Reason: The K-Ranking mechanism at audit/cited publication:325-329 awards beta at the reported rank exactly when sigma_v(i) is at most K, and zero otherwise. The actual target tests rankOf ranking candidate < K on zero-based ranks and returns beta at that rank or zero. Candidate expands to Fin(n+2), Ranking to a permutation, and rankOf to its inverse, so the cutoff and positional score coincide with the source on every admissible K-ranking instance.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.ProofBridge.paper_definition_design_invariant

Paper-local declaration:
GGSG19TopThree.ProofBridge.paper_definition_design_invariant
Source locator: cited publication, lines 376-384

Verbatim paper-source connection

```
\subsection{Asymptotic design invariance}
\label{sec:consistency}
The asymptotic outcome $O^*$ of an election may vary with the scoring rule $\beta$. For example, there may be a different winner if voters are asked to
↪  identify their favorite two candidates than if they identify their single favorite candidate, if the winner in the latter case is a polarizing candidate. As
↪  an axiomatic comparison between outcomes is out of the scope of this paper, we restrict our attention to cases where all ``reasonable'' choices of
↪  different $\beta$ asymptotically result in the same outcome (where ``reasonable'' corresponds to the set of scoring rules $\mB$ defined above).


\begin{definition}
    A setting $(M, F)$ is \textit{asymptotically design-invariant} for goal $G$ if any reasonable $\beta$ induces the same outcome asymptotically.
    ↪  $\exists O^*: \forall \beta \in \mB,$
    \[ \lim_{N \to \infty} O^N(M, F, \beta, G) = O^*, \text{ with probability } 1 \]
\end{definition}
```

Lean-expanded paper semantic target

```
type:
{Ω : Type u_1} →
  {Rule : Type u_2} →
    {Outcome : Type u_3} →
      [inst : MeasurableSpace Ω] → MeasureTheory.Measure Ω → (Rule → Prop) → (Rule → ℕ → Ω → Outcome) → Prop

value:
fun {Ω : Type u_1} {Rule : Type u_2} {Outcome : Type u_3} [MeasurableSpace Ω] (μ : MeasureTheory.Measure Ω)
    (reasonable : Rule → Prop) (outcome : Rule → ℕ → Ω → Outcome) =>
  ∃ (Ostar : Outcome), GGSG19TopThree.ProofBridge.paper_definition_design_invariant_for μ reasonable outcome Ostar
```

Direct retained dependencies

- GGSG19TopThree.ProofBridge.paper_definition_design_invariant_for

Recorded source-to-declaration screening **Verdict:** matches

Reason: Definition 1 at audit/cited publication:376-384 requires one outcome Ostar shared by every reasonable rule, with probability-one convergence for each rule. The actual predicate first existentially quantifies Ostar, then the displayed dependency quantifies every reasonable beta and almost every sample path, followed by eventual equality to Ostar. Elections have a finite outcome space, so convergence to that outcome is equivalent to eventual equality. The quantifier order and common-outcome requirement match; the generic rule, outcome, and measure parameters specialize to the paper election.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.ProofBridge.paper_definition_large_deviation_rate

Paper-local declaration:
GGSG19TopThree.ProofBridge.paper_definition_large_deviation_rate
Source locator: cited publication, lines 449-458

Verbatim paper-source connection

```
\subsection{Learning rates}

We begin by deriving rates for how quickly a given positional scoring rule $\beta$ learns its asymptotic outcome $O^*$ (given it exists), as a function of
↪  the voter preference model $F$. %
In particular, we use \textit{large deviation rates} at which a scoring rule learns~\cite{dembo_large_2010}.
\begin{definition}
    Consider a sequence $\{A_N \geq 0\}_{N \in \bbN}$, where $A_N \to 0$ . Value $r>0$ is the \textit{large deviation rate} for $A_N$ if
    \begin{align*}
    r = -\lim_{N\to\infty}\frac{1}{N} \log A_N
    \end{align*}
\end{definition} When $r>0$ exists, $A_N \to 0$ exponentially fast, with exponent $r$ asymptotically, i.e., $A_N$ is $e^{-rN \pm o(N)}$. These rates
↪  provide us both upper and lower bounds for the probability of an error or the number of errors in an outcome after $N$ voters, up to polynomial
↪  factors. In particular, in the propositions below, we will calculate the large deviation rate of errors in the outcome. We will also then provide (loose)
↪  upper bounds for such errors after $N$ voters that hold without any missing polynomial factors, for any $N$. These upper bounds are equivalent to
↪  Chernoff bounds.
```

Lean-expanded paper semantic target

```
type:
(ℕ → ℝ) → ℝ → Prop

value:
fun (A : ℕ → ℝ) (r : ℝ) =>
  (∀ (N : ℕ), 0 ≤ A N) ∧
    Filter.Tendsto A Filter.atTop (nhds 0) ∧
      0 < r ∧ Filter.Tendsto (fun (N : ℕ) => -(↑N)⁻¹ * Real.log (A N)) Filter.atTop (nhds r)
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: Definition 2 at audit/cited publication:449-458 defines a positive finite r for a nonnegative sequence tending to zero, with r equal to the limit of -log(A_N)/N. The actual predicate includes all four components: pointwise nonnegativity, convergence of A to zero, r>0, and convergence of the negative real reciprocal of N times Real.log(A N) to r. The N=0 totalization does not affect an atTop limit. A zero subsequence cannot yield a positive finite rate in either reading, so the totalized logarithm does not add a source-admissible finite-rate case.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

GGSG19TopThree.ProofBridge.paper_definition_rate_optimal

Paper-local declaration:
GGSG19TopThree.ProofBridge.paper_definition_rate_optimal
Source locator: cited publication, lines 542-553

Verbatim paper-source connection

```
\subsection{Optimal design and discussion}
Now that we can quantify how quickly a given scoring rule $\beta$ learns its asymptotic outcome, we apply our framework to \textit{designing} elections,
↪  i.e., choosing an optimal scoring rule $\beta$. For the rest of this work, we assume that the setting $(M,F)$ is asymptotically design-invariant for the
↪  goal $G$, i.e., there exists an outcome that is asymptotically induced by every reasonable scoring rule. Then, the design of an election $\beta$ only
↪  affects the \textit{rate} at which the election converges to the asymptotic outcome $O^*$, as calculated above. With no other constraints, then, the
↪  design challenge is simple: find the rate optimal $\beta$.
\begin{definition}
      A scoring rule $\beta^* \in \mB$ is \textit{rate optimal} if it maximizes the rate in Proposition~\ref{thm:goal_learning}. %
      %
      %
      $K^*$-Approval is \textit{Approval rate optimal} if it maximizes the rate among $K$-Approval mechanisms.
      %
      %
      %
\end{definition}
```

Lean-expanded paper semantic target

```
type:
{Rule : Type u_1} → Set Rule → (Rule → ℝ) → Rule → Prop

value:
fun {Rule : Type u_1} (feasible : Set Rule) (rate : Rule → ℝ) (βstar : Rule) =>
  βstar ∈ feasible ∧ ∀ β ∈ feasible, rate β ≤ rate βstar
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: Definition 3 at audit/cited publication:542-553 requires beta-star to belong to the reasonable design class and maximize the learning rate over that class; Approval rate optimality restricts that comparison to K-Approval mechanisms. The actual predicate is membership in the supplied feasible class together with rate beta <= rate beta-star for every feasible beta. Instantiating that class with either source design family gives precisely its corresponding definition. The rate function is a parameter, preserving the source reference to the previously defined rate rather than replacing it with a different objective.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.ReasonablePrefixWeights

Paper-local declaration:
GGSG19TopThree.ReasonablePrefixWeights
Source locator: cited publication, lines 339-342

Verbatim paper-source connection

β encodes both elicitation and aggregation. For example, K-Approval is equivalent to K-ranking with score function $\beta(k) = \bbI[k \leq K]$.
↪ Furthermore, note that given K-ranking data, one can simulate K'-ranking elicitation for $K'\leq K$ with a β s.t. $\beta(k) = 0$ for $k>K'$. %

The scoring rule β is a design choice made by the election organizer, and so we will refer to β as the election's \textit{design}. We restrict
↪ ourselves to non-constant, non-increasing scoring rules, i.e., $\beta \in \mB = \{\beta : \forall k<\ell \in {1,\dots,M}, \beta(k)\geq \beta(\ell), \text{ and
↪ } \exists k<\ell, \beta(k)> \beta(\ell) \}$.

Lean-expanded paper semantic target

type:
{Cut : Type u_1} → (Cut → ℝ) → Prop

value:
fun {Cut : Type u_1} (diff : Cut → ℝ) => (∀ (k : Cut), 0 ≤ diff k) ∧ ∃ (k : Cut), 0 < diff k

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source at audit/cited publication:339-342 restricts beta to nonincreasing, nonconstant positional scores. The actual predicate requires every adjacent-prefix coefficient to be nonnegative and at least one to be positive. In the displayed paper use the coefficient domain is exactly the proper prefixes Fin(n+1) of n+2 candidates. For a source rule take diff(k)=beta(k)-beta(k+1); conversely the displayed rankingPrefixScore reconstructs beta(rank)-beta(last). This common additive offset leaves all pairwise cumulative-score comparisons and uniform tie events unchanged. Thus the representation covers the full source class in its displayed outcome use.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain

Paper-local declaration:
GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain
Source locator: cited publication, lines 292-296

Verbatim paper-source connection

A special case for F is the \textit{Mallows} model \cite{mallows1957non}, in which there is a ``true'' societal preference σ^* from which each voter's ranking is a noisy sample. In particular,
\[
F_\text{Mallows}(\sigma) \propto {\phi^{d(\sigma, \sigma^*)}}
\]
Where $d(\sigma, \sigma^*)$ is the Kendall's τ distance between rankings σ, σ^*, and $\phi \in [0,1]$ is the noise parameter: the smaller it is, the more concentrated F is around σ^*.

[Next verbatim source excerpt]

\begin{restatable}{corollary}{lemmallowsnorando}
\label{lem:mallowsnorando}
Randomization among K-Approval mechanisms does not improve the learning rate for selecting W winners from the Mallows model. For any randomized K-Approval mechanism (B,D), where $\beta_p\in B$ corresponds to p-Approval, for selecting W winners from the Mallows model, there exists an Approval rate optimal mechanism K^*-Approval such that $r(K^*) \geq r(B,D)$.

\end{restatable}

The proof simply notes that under the Mallows model with this goal, the candidate pair $W, W+1$ (when candidates are indexed according to reference distribution σ^*) is pivotal regardless of the K-Approval mechanism used.

[Next verbatim source excerpt]

Recall that in a Mallows model, each voter's ranking is a noisy sample from a reference distribution σ^*. With this symmetric model, one may believe that setting $K=W$ is always optimal. For example, when noise parameter $\phi = 0$ and so each voter's ranking is exactly σ^*, $K=W$ is optimal; in fact, any other design $K \neq W$ fails to correctly identify the set of winners even asymptotically: it would not distinguish among the first K candidates in σ^* or among the last $M-K$ candidates. However, our next result establishes that the cases with $\phi > 0$ are different.

Lean-expanded paper semantic target

type:
{n : ℕ} → ℝ → AppliedModelingLib.SocialChoice.Ranking.Candidate n → Prop

value:
fun {n : ℕ} (q : ℝ) (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n) => 0 < ↑W ∧ 0 < q ∧ q < 1

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate

Recorded source-to-declaration screening **Verdict:** matches

Reason: For the positive-noise Mallows corollary, the source uses a nontrivial W-winner problem and distinguishes phi greater than zero from the deterministic endpoint. With Candidate n representing n+2 candidates, 0 < W.val and 0 < q < 1 are exactly the proper W and positive-noise domain.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain

Paper-local declaration:
GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain
Source locator: cited publication, lines 290-296

Verbatim paper-source connection

\parbold{Candidates and Voters} There is a set of M candidates $C = \{1,\dots,M\}$, typically indexed by $i,j\in C$. There are N voters $V = \{1,\dots,N\}$. Each voter $v\in V$ has a strict ranking of candidates σ_v, drawn independently and identically from probability mass function over strict rankings $F(\sigma)$. Let $i \succ_\sigma j$ denote that i is preferred over j in σ, and $\sigma(i) = k$ denote that candidate i is in the kth position in σ.

A special case for F is the \textit{Mallows} model \cite{mallows1957non}, in which there is a ``true'' societal preference σ^* from which each voter's ranking is a noisy sample. In particular,
\[
F_\text{Mallows}(\sigma) \propto {\phi^{d(\sigma, \sigma^*)}}
\]
Where $d(\sigma, \sigma^*)$ is the Kendall's τ distance between rankings σ, σ^*, and $\phi \in [0,1]$ is the noise parameter: the smaller it is, the more concentrated F is around σ^*.

[Next verbatim source excerpt]

Let the items in the reference ranking be, in order, $i=1, 2, 3, 4$, respectively.

A Mallows model (with parameter $\phi = \frac{p}{1-p}$, where p is the probability of flipping a given pair of candidates) can be sampled by repeated insertion \citep{lu14a,diaconis_group_nodate}: starting from the first item in the reference ranking, there exists probability $\tilde{p}_{ij} = \frac{\phi^{i-j}}{1 + \phi^1 + \dots + \phi^{i-1}}$ at which item i can be inserted into position $j\leq i$, \textit{independently} of how items above it were inserted, such that the resulting ranking distribution matches the Mallows model.

Using this repeated insertion property for our example, we can derive $p_{\ell k}$, the probability at which item 3 is in position ℓ and item 4 is in position k after sampling from a Mallows model with parameter ϕ.

In particular, if $\ell < k$, $p_{\ell k}$ is exactly the probability that item 3 is inserted in position ℓ and item 4 is inserted in position k. If $k>\ell$, however, it is the probability that item 3 is inserted in position $\ell - 1$ and then pushed down when item 4 is inserted in position k. (More generally, it turns out, the exact probability for an item appearing in a given position in the Mallows model can be calculated using a simple dynamic program, a fact that does not appear to be documented elsewhere but may be independently useful. We used this dynamic program to find this given example).

Lean-expanded paper semantic target

type:
$\mathbb{R} \to \text{Prop}$

value:
$\text{fun } (q : \mathbb{R}) \Rightarrow 0 \le q \wedge q \le 1$

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source Mallows definition prints the closed noise range phi in [0,1], including its deterministic endpoint. The repeated-insertion kernel uses exactly that closed q domain.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.assumption_nontrivial_k_approval_cutoffs

Paper-local declaration:
GGSG19TopThree.assumption_nontrivial_k_approval_cutoffs
Source locator: cited publication, lines 339-342

Verbatim paper-source connection

β encodes both elicitation and aggregation. For example, K-Approval is equivalent to K-ranking with score function $\beta(k) = \bbI[k \leq K]$.
↪ Furthermore, note that given K-ranking data, one can simulate K'-ranking elicitation for $K'\leq K$ with a β s.t. $\beta(k) = 0$ for $k>K'$. %

The scoring rule β is a design choice made by the election organizer, and so we will refer to β as the election's \textit{design}. We restrict
↪ ourselves to non-constant, non-increasing scoring rules, i.e., $\beta \in \mB = \{\beta : \forall k<\ell \in {1,\dots,M}, \beta(k)\geq \beta(\ell), \text{ and
↪ } \exists k<\ell, \beta(k)> \beta(\ell) \}$.

[Next verbatim source excerpt]

\begin{restatable}{theorem}{lemrandomizenotbetterapproval}
\label{lem:randomizenotbetterapproval}

Randomization amongst K-Approval mechanisms does not improve the learning rate for separating a given pair of candidates i,j for any
↪ asymptotically design-invariant noise model F or goal G. For any randomized K-Approval mechanism (B,D), where $\beta_p\in B$
↪ corresponds to p-Approval, for any F, G, there exists a mechanism K_{ij}^*-Approval such that $r_{ij}(K_{ij}^*) \geq r_{ij}(B,D)$.

%

%

%
%
%
\end{restatable}

[Next verbatim source excerpt]

The result follows by convexity. Consider a randomization of K-Approval mechanisms for $K\in\{1, \dots, M-1\}$, where K-Approval is used with
↪ probability d^K.

%

The resulting approval probabilities are: $t^i_{ij} = \sum_{K=1}^{M-1} d^K t^i_{ij}(K), t^j_{ij} = \sum_{K=1}^{M-1} d^K t^j_{ij}(K)$. By
↪ convexity:
\begin{align*}
r\left(\sum_{K=1}^{M-1} d^K t^i_{ij}(K), \sum_{K=1}^{M-1} d^K t^j_{ij}(K)\right) &\leq \sum_{K=1}^{M-1} d^K r\left(t^i_{ij}(K),
↪ t^j_{ij}(K)\right)\\
&= \sum_{K=1}^{M-1} d^K r_{ij}(K)\\
&\leq \max_K r_{ij}(K)

Lean-expanded paper semantic target

type:
{n : ℕ} → {Rule : Type u_1} → [Fintype Rule] → (Rule → ℕ) → Prop

value:
fun {n : ℕ} {Rule : Type u_1} [Fintype Rule] (K : Rule → ℕ) =>
(∀ (rule : Rule), 0 < K rule) ∧ ∀ (rule : Rule), K rule < n + 2

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source randomizes over K in {1,...,M-1}. Since Candidate n has M=n+2 elements, the target's 0 < K and K < n+2 are exactly the source proper, nonconstant K-Approval cutoffs.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.assumption_pairwise_approval_ternary_gap_domain

Paper-local declaration:
GGSG19TopThree.assumption_pairwise_approval_ternary_gap_domain
Source locator: cited publication, lines 508-516

Verbatim paper-source connection

```
\begin{restatable}{proposition}{lempairwiseapproval}
	\label{lem:pairwiselearning_approval}
	Consider $\beta$ consistent with $K$-Approval voting for some fixed $K$, and candidates $i,j$ such that $s_i > s_j$. Then the large deviation rate
	↪  $r_{ij}(\beta)$ in Proposition~\ref{thm:pairwiselearning} is
	\begin{align*}
	r_{ij}(K) &= -\log\left( 2\sqrt{t_{ij}^i(K)t_{ij}^j(K)} + 1 - t_{ij}^i(K) - t_{ij}^j(K) \right)
	\end{align*}
	Where $t^i_{ij}(K) \triangleq \pr_F(\sigma_v(i) \leq K, \sigma_v(j) > K)$, i.e., the probability that a voter approves $i$ but not $j$.
%
\end{restatable}
```

[Next verbatim source excerpt]

```
\lempairwiseapproval*
\begin{proof}
	With $K$-approval voting, $A^{ij}_v$ becomes
	\begin{align*}
	A_v = \begin{cases}
	1 & \text{w.p. } t^i_{ij}, \text{ i.e., when candidate $i$ approved but $j$ not approved}\\
	0 & \text{w.p. } 1-t^i_{ij} - t^j_{ij}, \text{ i.e., when both approved, or neither approved}\\
	-1 & \text{w.p. } t^j_{ij}, \text{ i.e., when candidate $j$ approved but $i$ not approved}
	\end{cases}
	\end{align*}
```

Lean-expanded paper semantic target

```
type:
{Signal : Type u_1} →
  [Fintype Signal] → [DecidableEq Signal] → PMF Signal → (Signal → ℝ) → (Signal → ℝ) → ℝ → ℝ → ℝ → Prop

value:
fun {Signal : Type u_1} [Fintype Signal] [DecidableEq Signal] (law : PMF Signal) (hiScore loScore : Signal → ℝ)
    (pUp pDown pZero : ℝ) =>
  pDown ≤ pUp ∧
    (∀ (signal : Signal),
       hiScore signal - loScore signal = 1 ∨
         hiScore signal - loScore signal = 0 ∨ hiScore signal - loScore signal = -1) ∧
      (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 1) = pUp ∧
        (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = -1) = pDown ∧
          (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 0) = pZero
```

Direct retained dependencies

- AppliedModelingLib.pmfProb

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source K-Approval proof has score gap +1 when only the higher candidate is approved, 0 when both or neither are approved, and -1 when only the lower candidate is approved. The target records those three exhaustive values, their PMF masses, and the source ordering pDown <= pUp.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.assumption_randomized_mechanism_probability_weights

Paper-local declaration:
GGSG19TopThree.assumption_randomized_mechanism_probability_weights
Source locator: cited publication, lines 666-670

Verbatim paper-source connection

```
\label{sec:randomization}

We now consider the question of whether \textit{randomizing} between mechanisms in an election may speed up learning. By randomization, we mean:
↪ consider a set of scoring rules $B = \{\beta_1, \dots, \beta_P\} \subseteq \mB$; elicitation and aggregation for a given voter is done according to a
↪ scoring rule picked at random from $B$, where $\beta_p$ is selected with probability $d_p$.

Note that the learning rate of such randomized schemes can be calculated as before, by summing across $\beta_p$ inside the $\bbE[\cdot]$ of
↪ $r_{ij}(\beta)$ or -- for $B$ consisting only of $K$-Approval votes -- directly through the resulting probability that the voter approves $i$ but not $j$.
↪ We use $r_{ij}(B,D), r(B,D)$ to denote the candidate pairwise and overall outcome learning rates, respectively, for randomized mechanism $(B,D)$,
↪ where $B = \{\beta_1, \dots, \beta_P\} \subseteq \mB$ and $D = \{d_1, \dots, d_P \}$.

[Next verbatim source excerpt]

    Similarly, if we use scoring rules $\{\beta^u\}_{u=1}^P$, each with probability $d^u$, then,
    \begin{align*}
    r_{ij}(\{\beta^u\}_{u=1}^P) &= - \log \inf_{z\in\bbR} \bbE_{F,\{d^u,\beta^u\}}\left[\exp\left[z\left[\beta^u(\sigma_v(i)) -
    ↪ \beta^u(\sigma_v(j))\right]\right]\right]\\
    &= - \log \inf_{z\in\bbR} \sum_u d^u  \bbE_{F}\left[\exp\left[z\left[\beta^u(\sigma_v(i)) - \beta^u(\sigma_v(j))\right]\right]\right]
    \end{align*}

[Next verbatim source excerpt]

    The result follows by convexity. Consider a randomization of $K$-Approval mechanisms for $K\in\{1, \dots, M-1\}$, where $K$-Approval is used with
    ↪ probability $d^K$.

%

    The resulting approval probabilities are: $t^i_{ij} = \sum_{K=1}^{M-1} d^K t^i_{ij}(K), t^j_{ij} = \sum_{K=1}^{M-1} d^K t^j_{ij}(K)$. By
    ↪ convexity:
    \begin{align*}
    r\left(\sum_{K=1}^{M-1} d^K t^i_{ij}(K), \sum_{K=1}^{M-1} d^K t^j_{ij}(K)\right) &\leq \sum_{K=1}^{M-1} d^K r\left(t^i_{ij}(K),
    ↪ t^j_{ij}(K)\right)\\
    &= \sum_{K=1}^{M-1} d^K  r_{ij}(K)\\
    &\leq \max_K r_{ij}(K)
```

Lean-expanded paper semantic target

```
type:
{Rule : Type u_1} → [Fintype Rule] → (Rule → ℝ) → Prop

value:
fun {Rule : Type u_1} [Fintype Rule] (weight : Rule → ℝ) =>
  (∀ (rule : Rule), 0 ≤ weight rule) ∧ ∑ rule : Rule, weight rule = 1
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source describes a finite family of scoring rules selected with probabilities d_p. The target's nonnegative finite weights summing to one is exactly that probability-vector condition.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.iidUniformTieTieredCorrect

Paper-local declaration:
GGSG19TopThree.iidUniformTieTieredCorrect
Source locator: cited publication, lines 345-345

Verbatim paper-source connection

```
\parbold{Outcome} After $N$ voters, candidate $i$'s cumulative score is $s_i^N = \frac{1}{N} \sum_{v=1}^N s_{iv}$. Candidates are ranked in
↪  descending order of score, to form ranking ${\sigma}^N$, with ties broken uniformly at random. We denote the \textit{outcome} after $N$ voters,
↪  corresponding to the goal $G$, as $O^N(M, F, \beta, G)$. For example, for the goal of selecting $W$ winners, $O^N(M, F, \beta, G)$ is simply the top
↪  $W$ candidates in ${\sigma}^N$. When $(M, F, \beta, G)$ is clear from context, we will refer to the outcome as $O^N$.
```

Lean-expanded paper semantic target

```
type:
{n Stage : ℕ} →
  (AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ) →
    (Fin Stage → Finset (AppliedModelingLib.SocialChoice.Ranking.Candidate n)) →
      (ℕ → AppliedModelingLib.SocialChoice.Ranking.Ranking n × AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
        ℕ → Prop

value:
fun {n Stage : ℕ}
    (score :
      AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ)
    (targetPrefix : Fin Stage → Finset (AppliedModelingLib.SocialChoice.Ranking.Candidate n))
    (path : ℕ → AppliedModelingLib.SocialChoice.Ranking.Ranking n × AppliedModelingLib.SocialChoice.Ranking.Ranking n)
    (N : ℕ) =>
  ∀ (stage : Fin Stage) (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n),
    hi ∈ targetPrefix stage →
      lo ∉ targetPrefix stage →
        GGSG19TopThree.iidUniformTieCandidateScore score path N lo <
            GGSG19TopThree.iidUniformTieCandidateScore score path N hi ∨
          GGSG19TopThree.iidUniformTieCandidateScore score path N lo =
              GGSG19TopThree.iidUniformTieCandidateScore score path N hi ∧
            AppliedModelingLib.SocialChoice.Ranking.rankOf (path N).2 hi <
              AppliedModelingLib.SocialChoice.Ranking.rankOf (path N).2 lo
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.rankOf
- GGSG19TopThree.iidUniformTieCandidateScore

Recorded source-to-declaration screening **Verdict:** matches

Reason: The outcome definition at audit/cited publication:345 ranks candidates by descending cumulative score, breaks ties uniformly, and takes the goal-specific top prefixes. The actual predicate requires each candidate inside every target prefix to precede every outside candidate, either by strictly greater total score or by equal total score and earlier rank in the supplied tie permutation. Its score dependency sums the first N voter scores. For positive N these comparisons equal comparisons of the source averages. The displayed PropositionOneUniformTieConsistency use supplies the independent uniform tie law and tests these prefix events eventually almost surely, so the predicate is exactly tiered outcome correctness.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.mallowsPairDPTabulationRun

Paper-local declaration:
GGSG19TopThree.mallowsPairDPTabulationRun
Source locator: cited publication, lines 763

Verbatim paper-source connection

We prove the theorem by example. To find this example and to generate the plots discussed next, we use an efficient dynamic program to exactly calculate the joint distributions of the locations $\sigma_v(i),\sigma_v(j)$ of pairs of candidates i,j in a voter's ranking, given the Mallows noise parameter; we can then directly calculate $t_{ij}^i(K), t_{ij}^j(K)$ and thus the learning rate for each K-Approval mechanism. This program leverages Mallows repeated insertion probabilities \cite{lu14a,diaconis_group_nodate} and may be of independent interest for numerical analyses of the Mallows model.

[Next verbatim source excerpt]

A Mallows model (with parameter $\phi = \frac{p}{1-p}$, where p is the probability of flipping a given pair of candidates) can be sampled by repeated insertion \citep{lu14a,diaconis_group_nodate}: starting from the first item in the reference ranking, there exists probability $\tilde{p}_{ij} = \frac{\phi^{i-j}}{1 + \phi^1 + \dots + \phi^{i-1}}$ at which item i can be inserted into position $j\leq i$, \textit{independently} of how items above it were inserted, such that the resulting ranking distribution matches the Mallows model.

Using this repeated insertion property for our example, we can derive $p_{\ell k}$, the probability at which item 3 is in position ℓ and item 4 is in position k after sampling from a Mallows model with parameter ϕ.

In particular, if $\ell < k$, $p_{\ell k}$ is exactly the probability that item 3 is inserted in position ℓ and item 4 is inserted in position k. If $k>\ell$, however, it is the probability that item 3 is inserted in position $\ell - 1$ and then pushed down when item 4 is inserted in position k. (More generally, it turns out, the exact probability for an item appearing in a given position in the Mallows model can be calculated using a simple dynamic program, a fact that does not appear to be documented elsewhere but may be independently useful. We used this dynamic program to find this given example).

Lean-expanded paper semantic target

type:
```
{N : ℕ} →
  GGSG19TopThree.MallowsRepeatedInsertionModel N → Fin N → Fin N → ℕ → GGSG19TopThree.MallowsPairDPTabulationResult N
```

value:
```
fun {N : ℕ} (model : GGSG19TopThree.MallowsRepeatedInsertionModel N) (leftStage rightStage : Fin N) (a : ℕ) =>
  Nat.brecOn a (GGSG19TopThree.mallowsPairDPTabulationRun._f model leftStage rightStage)
```

Direct retained dependencies

- GGSG19TopThree.MallowsPairDPTabulationResult
- GGSG19TopThree.MallowsPairPositionState
- GGSG19TopThree.MallowsRepeatedInsertionModel
- GGSG19TopThree.mallowsPairExplicitTableStep
- GGSG19TopThree.mallowsPairInitialState

Recorded source-to-declaration screening **Verdict:** matches

Reason: The pinned source at audit/cited publication:763 and 1546-1550 describes an exact pair-location dynamic program using independent repeated insertions with weight phi^(i-j)/(1+...+phi^(i-1)). The displayed insertion model gives precisely this formula in zero-based indices. Starting from mass one on (none,none), the actual runner tabulates each successive insertion: transition mass sums insertion probabilities producing the target state, and the next table sums prior mass times transition mass. A tracked candidate acquires its insertion location at its own stage and is shifted when a later insertion is weakly above it. These displayed bodies compute the joint positions for arbitrary tracked candidates after all N stages; extra steps leave the completed table fixed. The bookkeeping count does not alter probabilities. The q in [0,1] domain agrees with the source Mallows definition at line 296.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.model_arbitrary_ordered_tier_goal

Paper-local declaration:
GGSG19TopThree.model_arbitrary_ordered_tier_goal
Source locator: cited publication, lines 309-312

Verbatim paper-source connection

\parbold{Election goal} We assume that the \textit{goal} G is to divide the candidates into T disjoint, ordered \textit{tiers} $G = \{C_1, \dots , C_T\}$, such that $C = \cup_{t=1}^T C_t$, where candidate $i \in C_{s}$ is deemed societally preferable over $j\in C_{t}$ if $s<t$. The size of each tier is fixed before the election. For example, recovering a strict ranking over all candidates corresponds to $G = \{C_1, \dots, C_M\}$, where $|C_t| = 1$. Alternatively, identifying a set of W winners, without distinguishing amongst the winners, corresponds to $G = \{C_1, C_2\}$, with $|C_1| = W$.

In the main text and especially the empirics, we will focus on the task of selecting W winners as it is the most common task in practice. However, this general notation allows comparison of the learning properties of different settings, and for example ask how much more expensive is it (in terms of the number of voters needed) to identify a strict ranking as opposed to just a set of winners.

Lean-expanded paper semantic target

type:
```
{Candidate : Type u_1} →
  [Fintype Candidate] → [DecidableEq Candidate] → (Stage : ℕ) → (Fin Stage → ℕ) → (Fin Stage → Finset Candidate) → Prop
```

value:
```
fun {Candidate : Type u_1} [Fintype Candidate] [DecidableEq Candidate] (Stage : ℕ) (tierSize : Fin Stage → ℕ)
    (tier : Fin Stage → Finset Candidate) =>
  (∀ (candidate : Candidate), ∃! stage : Fin Stage, candidate ∈ tier stage) ∧
    ∀ (stage : Fin Stage), (tier stage).card = tierSize stage
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source goal is a finite ordered partition of all candidates with sizes fixed before the election. Unique tier membership gives disjoint exhaustive tiers, and the cardinality equation records the declared fixed size of each tier.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.model_finite_iid_strict_rankings

Paper-local declaration:
GGSG19TopThree.model_finite_iid_strict_rankings
Source locator: cited publication, lines 286-296

Verbatim paper-source connection

```
\subsection{Model primitives}

We begin with the model primitives: candidates and voters, the election goal, and elicitation and aggregation.

\parbold{Candidates and Voters} There is a set of $M$ candidates $C = \{1,\dots,M\}$, typically indexed by $i,j\in C$. There are $N$ voters $V =
↪  \{1,\dots,N\}$. Each voter $v\in V$ has a strict ranking of candidates $\sigma_v$, drawn independently and identically from probability mass function
↪  over strict rankings $F(\sigma)$. Let $i \succ_\sigma j$ denote that $i$ is preferred over $j$ in $\sigma$, and $\sigma(i) = k$ denote that candidate
↪  $i$ is in the $k$th position in $\sigma$.

A special case for $F$ is the \textit{Mallows} model \cite{mallows1957non}, in which there is a ``true'' societal preference $\sigma^*$ from which each
↪  voter's ranking is a noisy sample. In particular,
\[
F_\text{Mallows}(\sigma) \propto {\phi^{d(\sigma, \sigma^*)}}
\]
Where $d(\sigma, \sigma^*)$ is the Kendall's $\tau$ distance between rankings $\sigma, \sigma^*$, and $\phi \in [0,1]$ is the noise parameter: the
↪  smaller it is, the more concentrated $F$ is around $\sigma^*$.
```

Lean-expanded paper semantic target

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    MeasureTheory.Measure (ℕ → AppliedModelingLib.SocialChoice.Ranking.Ranking n)

value:
fun {n : ℕ} (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)) =>
  AppliedModelingLib.finitePMFIidPathMeasure law
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.instMeasurableSpaceRanking
- AppliedModelingLib.finitePMFIidPathMeasure

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source draws every voter independently and identically from one PMF on strict rankings. The target is the canonical infinite iid path measure for that PMF, whose finite initial segments are the stated elections.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.model_mallows_ranking_mass_and_parameter_domain

Paper-local declaration:
GGSG19TopThree.model_mallows_ranking_mass_and_parameter_domain
Source locator: cited publication, lines 292-296

Verbatim paper-source connection

A special case for F is the \textit{Mallows} model \cite{mallows1957non}, in which there is a ``true'' societal preference σ^* from which each voter's ranking is a noisy sample. In particular,
\[
F_\text{Mallows}(\sigma) \propto {\phi^{d(\sigma, \sigma^*)}}
\]
Where $d(\sigma, \sigma^*)$ is the Kendall's τ distance between rankings σ, σ^*, and $\phi \in [0,1]$ is the noise parameter: the smaller it is, the more concentrated F is around σ^*.

Lean-expanded paper semantic target

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ → Prop

value:
fun {n : ℕ} (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n))
    (center : AppliedModelingLib.SocialChoice.Ranking.Ranking n) (q : ℝ) =>
  0 ≤ q ∧
    q ≤ 1 ∧
      ∃ (normalizer : ℝ),
        0 < normalizer ∧
          ∀ (ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n),
            (law ranking).toReal = AppliedModelingLib.SocialChoice.Ranking.mallowsWeight q center ranking / normalizer
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.mallowsWeight

Recorded source-to-declaration screening **Verdict:** matches

Reason: The target gives the printed Mallows content: a strict-ranking PMF whose mass is proportional to q raised to Kendall--tau distance from a reference ranking, under 0 <= q <= 1. A positive normalizer is the explicit meaning of the source proportionality symbol.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.model_randomized_scoring_mechanism

Paper-local declaration:
GGSG19TopThree.model_randomized_scoring_mechanism
Source locator: cited publication, lines 662-670

Verbatim paper-source connection

```
\subsection{When does randomization help?}
%

%
\label{sec:randomization}

We now consider the question of whether \textit{randomizing} between mechanisms in an election may speed up learning. By randomization, we mean:
↪ consider a set of scoring rules $B = \{\beta_1, \dots, \beta_P\} \subseteq \mB$; elicitation and aggregation for a given voter is done according to a
↪ scoring rule picked at random from $B$, where $\beta_p$ is selected with probability $d_p$.

Note that the learning rate of such randomized schemes can be calculated as before, by summing across $\beta_p$ inside the $\bbE[\cdot]$ of
↪ $r_{ij}(\beta)$ or -- for $B$ consisting only of $K$-Approval votes -- directly through the resulting probability that the voter approves $i$ but not $j$.
↪ We use $r_{ij}(B,D), r(B,D)$ to denote the candidate pairwise and overall outcome learning rates, respectively, for randomized mechanism $(B,D)$,
↪ where $B = \{\beta_1, \dots, \beta_P\} \subseteq \mB$ and $D = \{d_1, \dots, d_P \}$.
```

Lean-expanded paper semantic target

```
type:
{n : ℕ} →
  {Rule : Type u_1} →
    [inst : Fintype Rule] →
      [DecidableEq Rule] →
        PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
          (Rule →
             AppliedModelingLib.SocialChoice.Ranking.Candidate n →
               AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ) →
            (weight : Rule → ℝ) →
              (∀ (rule : Rule), 0 ≤ weight rule) →
                ∑ rule : Rule, weight rule = 1 →
                  PMF (Rule × AppliedModelingLib.SocialChoice.Ranking.Ranking n) ×
                    (AppliedModelingLib.SocialChoice.Ranking.Candidate n →
                      Rule × AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ)

value:
fun {n : ℕ} {Rule : Type u_1} [Fintype Rule] [DecidableEq Rule]
    (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n))
    (score :
      Rule →
        AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ)
    (weight : Rule → ℝ) (hweight : ∀ (rule : Rule), 0 ≤ weight rule) (hsum : ∑ rule : Rule, weight rule = 1) =>
  (GGSG19TopThree.randomizedScoringSamplingLaw law weight hweight hsum,
    fun (candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n)
      (draw : Rule × AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
    score draw.1 candidate draw.2)
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.randomizedScoringSamplingLaw

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source mechanism first randomly selects one score rule from the finite probability vector and applies it to a sampled strict ranking. The target constructs that joint sampling PMF and the corresponding indexed score evaluator, making the two draws and selected rule explicit.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.rankingPrefixScore

Paper-local declaration:
GGSG19TopThree.rankingPrefixScore
Source locator: cited publication, lines 339-342

Verbatim paper-source connection

β encodes both elicitation and aggregation. For example, K-Approval is equivalent to K-ranking with score function $\beta(k) = \bbI[k \leq K]$.
↪ Furthermore, note that given K-ranking data, one can simulate K'-ranking elicitation for $K'\leq K$ with a β s.t. $\beta(k) = 0$ for $k>K'$. %

The scoring rule β is a design choice made by the election organizer, and so we will refer to β as the election's \textit{design}. We restrict
↪ ourselves to non-constant, non-increasing scoring rules, i.e., $\beta \in \mB = \{\beta : \forall k<\ell \in {1,\dots,M}, \beta(k)\geq \beta(\ell), \text{ and
↪ } \exists k<\ell, \beta(k)> \beta(\ell) \}$.

Lean-expanded paper semantic target

type:
{n : ℕ} →
(GGSG19TopThree.RankingProperPrefixCut n → ℝ) →
AppliedModelingLib.SocialChoice.Ranking.Candidate n → AppliedModelingLib.SocialChoice.Ranking.Ranking n → ℝ

value:
fun {n : ℕ} (diff : GGSG19TopThree.RankingProperPrefixCut n → ℝ)
(candidate : AppliedModelingLib.SocialChoice.Ranking.Candidate n)
(ranking : AppliedModelingLib.SocialChoice.Ranking.Ranking n) =>
GGSG19TopThree.prefixScoreFromEvent diff GGSG19TopThree.rankingInTopPrefix candidate ranking

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.RankingProperPrefixCut
- GGSG19TopThree.prefixScoreFromEvent
- GGSG19TopThree.rankingInTopPrefix
- GGSG19TopThree.rankingInTopPrefix_decidable

Recorded source-to-declaration screening **Verdict:** matches

Reason: The reasonable-score model at audit/cited publication:339-342 consists of nonconstant nonincreasing positional beta. The actual score is the displayed finite sum over proper prefixes of diff(cut) times the indicator rank<=cut. Substituting adjacent differences beta(k)-beta(k+1) telescopes to beta(rank)-beta(last), and the displayed ReasonablePrefixWeights condition is exactly nonnegativity with one strict drop. All source score vectors are represented up to that common additive offset. The displayed paper use compares cumulative candidate scores and ties, both invariant under this offset, so the actual function preserves the source election behavior throughout that use.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GGSG19TopThree.rankingUniformTieSignalLaw

Paper-local declaration:
GGSG19TopThree.rankingUniformTieSignalLaw
Source locator: cited publication, lines 345-345

Verbatim paper-source connection

```
\parbold{Outcome} After $N$ voters, candidate $i$'s cumulative score is $s_i^N = \frac{1}{N} \sum_{v=1}^N s_{iv}$. Candidates are ranked in
↪  descending order of score, to form ranking ${\sigma}^N$, with ties broken uniformly at random. We denote the \textit{outcome} after $N$ voters,
↪  corresponding to the goal $G$, as $O^N(M, F, \beta, G)$. For example, for the goal of selecting $W$ winners, $O^N(M, F, \beta, G)$ is simply the top
↪  $W$ candidates in ${\sigma}^N$. When $(M, F, \beta, G)$ is clear from context, we will refer to the outcome as $O^N$.
```

Lean-expanded paper semantic target

```
type:
{n : ℕ} →
  PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n) →
    PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n × AppliedModelingLib.SocialChoice.Ranking.Ranking n)

value:
fun {n : ℕ} (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)) =>
  AppliedModelingLib.pmfProd law (PMF.uniformOfFintype (AppliedModelingLib.SocialChoice.Ranking.Ranking n))
```

Direct retained dependencies

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.pmfProd

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source outcome paragraph at audit/cited publication:345 requires uniform random tie breaking after descending-score ranking. The actual function forms the product of the voter-ranking PMF with the uniform PMF on all candidate permutations. Its displayed pmfProd body multiplies the two masses, giving independence without changing the voter law. The displayed tier-correctness predicate uses the second-coordinate permutation only when cumulative scores tie, and PropositionOneUniformTieConsistency supplies iid draws of this product law. Restricting a uniform permutation to each tied block gives the required uniform tie ordering.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Source claims

Review row 1

Source locator: cited publication, lines 386-391

Verbatim source input

Such design invariance only occurs under a fairly strong condition on the voter preference distribution: that the candidates can be separated into tiers (according to goal G) such that candidates in higher tiers are \textit{strictly} more likely to be ranked by a voter in the top k positions, for all $k<M$, than are candidates in lower tiers.

\begin{restatable}{proposition}{thmconsistency}
\label{thm:consistency}
A setting (M, F) for goal G is asymptotically design-invariant if and only if there exist candidate tiers $O^* = \{C_1^*\dots C_T^*\}$ (corresponding to G) s.t. $\forall s<t$: $i \in C^*_s, j\in C^*_t \implies$ $\pr_F(\sigma_v(i) \leq k) > \pr_F(\sigma_v(j) \leq k)$, $\forall k\in\{1 \dots M-1 \}$.
\end{restatable}

[Next verbatim source excerpt]

\thmconsistency*
%
\begin{proof}
\begin{align*}
%
\forall i \in C: \bbE[s_{iv}]
&= \sum_{m = 1}^M \beta(m) \pr_F(\sigma_v(i) = m)\\
&= \sum_{m = 1}^M \beta(m) \pr_F(\sigma_v(i) \leq m) - \sum_{m = 2}^M \beta(m) \pr_F(\sigma_v(i) < m) \\
&= \sum_{m = 1}^M \beta(m) \pr_F(\sigma_v(i) \leq m) - \sum_{m = 1}^{M-1} \beta(m+1) \pr_F(\sigma_v(i) \leq m) \\
&= \beta(M) + \sum_{m = 1}^{M-1} \left[\beta(m)- \beta(m+1)\right] \pr_F(\sigma_v(i) \leq m)
\end{align*}
~\\\\ $\implies$. By the definition of asymptotically design-invariant, $$ \exists O^*: \forall \beta \in \mB, \lim_{N \to \infty} \mO(M, N, F, \beta, G) = O^*, \text{ with probability } 1 $$
For this $O^* = \{C_1^*, \dots, C_T^*\}$, we show by contradiction that $\forall s<t$: $i \in C^*_s, j\in C^*_t \implies$ $\pr_F(\sigma_v(i) \leq k) > \pr_F(\sigma_v(j) \leq k)$, $\forall k\in\{1 \dots M-1 \}$: Suppose $\exists i \in C^*_s, j\in C^*_t, s<t, k \in \{1 \dots M-1 \}$ such that $\pr_F(\sigma_v(i)\leq k) \leq \pr_F(\sigma_v(j)\leq k)$. Then, let
$$\beta(m) = \begin{cases} 1 & m \leq k \\
0 & m > k
\end{cases}$$
Then, $\bbE[s_{iv}] = \beta(M) + \beta(k)\pr_F(\sigma_v(i)\leq k) \leq \bbE[s_{jv}]$. Then, with positive probability, $$\lim_{N \to \infty} \mO(M, N, F, \beta, G) \neq O^*$$.
%

~\\$\impliedby$. Suppose there exists such a O^*. Then, $\forall \beta \in \mB = \{\beta : \forall k<\ell \in {1 \dots M}, \beta(k)\geq \beta(\ell), \text{ and } \exists k<\ell, \beta(k)> \beta(\ell) \}$:
Suppose $i\in C_s^*, j \in C_t^*, s < t$:
\begin{align*}
\bbE[s_{iv}] &= \beta(M) + \sum_{m = 1}^{M-1} \left[\beta(m)- \beta(m+1)\right] \pr_F(\sigma_v(i) \leq m)\\
&> \bbE[s_{jv}]
\end{align*}
Where the strict inequality follows as $\exists m: \beta(m) - \beta(m+1)>0$. Then, for all candidates $i\in C_s^*, j \in C_t^*, s < t$, by the strong law of large numbers $\lim_{N\to\infty} s_i^N > \lim_{N\to\infty} s_j^N$ w.p. 1. Thus, $\lim_{N \to \infty} \mO(M, N, F, \beta, G) = O^*$ w.p. 1.

Expanded PaperInterface specification

```
∀ {n Stage : ℕ} (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n))
  (targetPrefix : Fin Stage → Finset (AppliedModelingLib.SocialChoice.Ranking.Candidate n)),
  (∀ (stage : Fin Stage) (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n),
     hi ∈ targetPrefix stage →
       lo ∉ targetPrefix stage →
         ∀ (cut : GGSG19TopThree.RankingProperPrefixCut n),
           GGSG19TopThree.rankingTopPrefixProb law lo cut < GGSG19TopThree.rankingTopPrefixProb law hi cut) ↔
    GGSG19TopThree.PropositionOneUniformTieConsistency law targetPrefix
```

Retained governing declarations

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.PropositionOneUniformTieConsistency
- GGSG19TopThree.RankingProperPrefixCut
- GGSG19TopThree.rankingTopPrefixProb

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: ℕ
3. parameter: PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)
4. parameter: Fin Stage → Finset (AppliedModelingLib.SocialChoice.Ranking.Candidate n)
5. conclusion: (∀ (stage : Fin Stage) (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n),
   hi ∈ targetPrefix stage →

```
    lo ∉ targetPrefix stage →
      ∀ (cut : GGSG19TopThree.RankingProperPrefixCut n),
        GGSG19TopThree.rankingTopPrefixProb law lo cut < GGSG19TopThree.rankingTopPrefixProb law hi cut) ↔
  GGSG19TopThree.PropositionOneUniformTieConsistency law targetPrefix
```

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proposition1_thm_consistency_tiered

Recorded source-to-Spec screening **Verdict:** matches

Reason: The target is the source iff: every higher-tier candidate has strictly greater top-prefix probability than each lower-tier candidate for every proper prefix exactly when the uniform-tie procedure is design invariant. It retains the reverse-direction equality case rather than adding a strict premise there.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 2

Source locator: cited publication, lines 471-485

Verbatim source input

```
\parbold{Rates for separating two candidates} We now derive the large deviation learning rates for recovering the true ordering between a pair of
↪  candidates $i,j$, given noise model $F$. These rates will directly translate to the learning rate for the overall election, given some goal $G$.


%

\begin{restatable}{proposition}{thmpairwiselearning}
    \label{thm:pairwiselearning}
    Fix scoring rule $\beta\in\mB$, voter distribution $F$, and consider candidates $i,j$ such that $s_i > s_j$. Then, the probability of making a mistake in
    ↪  ranking these two candidates after $N$ voters, $\pr(\sigma^N(i)>\sigma^N(j))$, goes to zero with large deviation rate
    \[r_{ij}(\beta) = -\inf_{z\in\bbR} \log \bbE_F\left[\exp\left(z\left(\beta(\sigma_v(i)) - \beta(\sigma_v(j))\right)\right)\right]\]
    Further, the following upper bound holds for any $N$.
    \[\pr(\sigma^N(i)>\sigma^N(j)) \leq \exp(-r_{ij}(\beta)N)\]
%
%
%
\end{restatable}

[Next verbatim source excerpt]

\thmpairwiselearning*
\begin{proof}

    Define the following random variable for each voter $v \sim F$:
    \begin{align*}
    A_v = \beta(\sigma_v(i)) - \beta(\sigma_v(j))
    \end{align*}
    Then, $\sigma^N(i) < \sigma^N(j)$ when $A^N = \sum_{v=1}^N A_v > 0$, and $\bbE[A_v]>0$ by supposition.
    Let
    \begin{align*}
        r_{ij}(\beta) &= -\inf_{z\in\bbR}{\Lambda(z)}\\
    \Lambda(z) &= \log \left[\sum_{m=1}^M \sum_{\ell\neq m} \pr(\sigma_v(i) = m, \sigma_v(j)=\ell) \exp[ z(\beta(\sigma_v(i)) - \beta(\sigma_v(j)))]
    ↪  \right]
    \end{align*}

    Then, by basic large deviation bounds (see, e.g.~\citet{dembo_large_2010}):
    \begin{align*}
    -\lim_{N\to \infty} \frac{1}{N} \log \pr(A^N \leq 0) &= r_{ij}(\beta)\\
%
    \end{align*}


    And, applying Chernoff bounds, we get the standard relationship to the large deviation rate, giving an upper bound for the probability of error
    ↪  directly, including any polynomial factors out front:
        \begin{align*}
    \pr(\sigma^N(i) > \sigma^N(j)) &\leq \pr(A^N \leq 0) \\
    &< \left[\inf_{z>0}\bbE[\exp[-zA_v]]\right]^N\\
%
    &= \left[\inf_{z<0}\left[\sum_{m=1}^M \sum_{\ell\neq m} \pr(\sigma_v(i) = m, \sigma_v(j)=\ell) \exp[ z(\beta(\sigma_v(i)) -
    ↪  \beta(\sigma_v(j)))]\right]\right]^N\\
        &= \left[\inf_{z<0} \exp[\Lambda(z)]\right]^N\\
        &= \exp[-r_{ij}(\beta)N]
    \end{align*}

Then, $\pr(\sigma^N(i) < \sigma^N(j)) > 1-\epsilon$ when
\begin{align*}
\exp[-r_{ij}N] &< \epsilon\\
\iff N&> \frac{1}{r_{ij}}\log\left(\frac{1}{\epsilon}\right)
\end{align*}

\end{proof}
```

Expanded PaperInterface specification

```
∀ {Signal : Type u_1} [inst : Fintype Signal] [inst_1 : DecidableEq Signal] (law : PMF Signal)
  (hiScore loScore : Signal → ℝ),
  (0 ≤ AppliedModelingLib.pmfExp law fun (signal : Signal) => hiScore signal - loScore signal) →
    (AppliedModelingLib.Probability.ExponentialRateCertificate
        (GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore)
        (GGSG19TopThree.pairwiseScoringRate law hiScore loScore) ∨
      (∃ (pZero : ℝ),
        (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 0) = pZero ∧
          0 < pZero ∧
            AppliedModelingLib.Probability.ExponentialRateCertificate
              (GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore) (-Real.log pZero)) ∨
      ∀ᶠ (n : ℕ) in Filter.atTop, GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore n = 0) ∧
    ∀ (n : ℕ),
      GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore n ≤
        Real.exp (-↑n * GGSG19TopThree.pairwiseScoringRate law hiScore loScore)
```

Retained governing declarations

- AppliedModelingLib.Probability.ExponentialRateCertificate

- AppliedModelingLib.pmfExp
- AppliedModelingLib.pmfProb
- GGSG19TopThree.pairwiseScoringErrorProb
- GGSG19TopThree.pairwiseScoringRate

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype Signal
3. parameter: DecidableEq Signal
4. parameter: PMF Signal
5. parameter: Signal → ℝ
6. parameter: Signal → ℝ
7. assumption: 0 ≤ AppliedModelingLib.pmfExp law fun (signal : Signal) => hiScore signal - loScore signal
8. conclusion: (AppliedModelingLib.Probability.ExponentialRateCertificate (GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore)
(GGSG19TopThree.pairwiseScoringRate law hiScore loScore) ∨
(∃ (pZero : ℝ),
(AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 0) = pZero ∧
0 < pZero ∧
AppliedModelingLib.Probability.ExponentialRateCertificate
(GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore) (-Real.log pZero)) ∨
∀ᶠ (n : ℕ) in Filter.atTop, GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore n = 0) ∧
∀ (n : ℕ),
GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore n ≤
Real.exp (-↑n * GGSG19TopThree.pairwiseScoringRate law hiScore loScore)

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proposition2_thm_pairwiselearning_finite_support

Recorded source-to-Spec screening **Verdict:** matches

Reason: The finite-support target has the source mean condition, Chernoff rate, and every-N upper bound. Its explicit zero-gap and strictly-positive-gap branches state the source formula at the one-sided boundaries.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 3

Source locator: cited publication, lines 498-516

Verbatim source input

For K-Approval voting, further, the rate simplifies.

%

\begin{restatable}{proposition}{lempairwiseapproval}
\label{lem:pairwiselearning_approval}
Consider β consistent with K-Approval voting for some fixed K, and candidates i,j such that $s_i > s_j$. Then the large deviation rate $r_{ij}(\beta)$ in Proposition~\ref{thm:pairwiselearning} is
\begin{align*}
r_{ij}(K) &= -\log\left(2\sqrt{t_{ij}^i(K)t_{ij}^j(K)} + 1 - t_{ij}^i(K) - t_{ij}^j(K) \right)
\end{align*}
Where $t^i_{ij}(K) \triangleq \pr_F(\sigma_v(i) \leq K, \sigma_v(j) > K)$, i.e., the probability that a voter approves i but not j.
%
\end{restatable}

[Next verbatim source excerpt]

\lempairwiseapproval*
\begin{proof}
With K-approval voting, A^{ij}_v becomes
\begin{align*}
A_v = \begin{cases}
1 & \text{w.p. } t^i_{ij}, \text{ i.e., when candidate i approved but j not approved}\\
0 & \text{w.p. } 1-t^i_{ij} - t^j_{ij}, \text{ i.e., when both approved, or neither approved}\\
-1 & \text{w.p. } t^j_{ij}, \text{ i.e., when candidate j approved but i not approved}
\end{cases}
\end{align*}
Then,
\begin{align*}
r_{ij} &= -\inf_{z\in\bbR}{\Lambda(z)}\\
\Lambda(z) &= \log \left[\sum_{m=1}^M \sum_{\ell\neq m} \pr(\sigma_v(i) = m, \sigma_v(j)=\ell) \exp[z(\beta(\sigma_v(i)) - \beta(\sigma_v(j)))] \right]\\
&= \log \left[t_{ij}^i \exp(z) + t_{ij}^j \exp(-z) + (1 - t_{ij}^i - t_{ij}^j)\right]
\end{align*}
The $\inf(\Lambda)$ is attained at $z = \frac{1}{2} \log\frac{t_{ij}^j}{t_{ij}^i}$ (Λ is convex in z, and so setting the first derivative to zero finds the $\inf$).
And so
\begin{align*}
r_{ij} &= -\log\left[t_{ij}^i \exp\left(\frac{1}{2} \log\frac{t_{ij}^j}{t_{ij}^i}\right) + t_{ij}^j \exp\left(-\frac{1}{2} \log\frac{t_{ij}^j}{t_{ij}^i}\right) + (1 - t_{ij}^i - t_{ij}^j) \right]\\
&= -\log\left[2\sqrt{t_{ij}^it_{ij}^j} + 1 - t_{ij}^i - t_{ij}^j \right]
\end{align*}
\end{proof}

Expanded PaperInterface specification

```
∀ {Signal : Type u_1} [inst : Fintype Signal] [inst_1 : DecidableEq Signal] (law : PMF Signal)
  (hiScore loScore : Signal → ℝ) {pUp pDown pZero : ℝ},
  pDown ≤ pUp →
    (∀ (signal : Signal),
       hiScore signal - loScore signal = 1 ∨
         hiScore signal - loScore signal = 0 ∨ hiScore signal - loScore signal = -1) →
      (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 1) = pUp →
        (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = -1) = pDown →
          (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 0) = pZero →
            AppliedModelingLib.Probability.ExponentialRateCertificate
              (GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore)
              (GGSG19TopThree.approvalPairwiseRate pUp pDown) ∨
            ∀ᶠ (n : ℕ) in Filter.atTop, GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore n = 0
```

Retained governing declarations

- AppliedModelingLib.Probability.ExponentialRateCertificate
- AppliedModelingLib.pmfProb
- GGSG19TopThree.approvalPairwiseRate
- GGSG19TopThree.pairwiseScoringErrorProb

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype Signal
3. parameter: DecidableEq Signal
4. parameter: PMF Signal

5. parameter: Signal → ℝ
6. parameter: Signal → ℝ
7. parameter: ℝ
8. parameter: ℝ
9. parameter: ℝ
10. assumption: pDown ≤ pUp
11. assumption: ∀ (signal : Signal),
  hiScore signal - loScore signal = 1 ∨ hiScore signal - loScore signal = 0 ∨ hiScore signal - loScore signal = -1
12. assumption: (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 1) = pUp
13. assumption: (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = -1) = pDown
14. assumption: (AppliedModelingLib.pmfProb law fun (signal : Signal) => hiScore signal - loScore signal = 0) = pZero
15. conclusion: AppliedModelingLib.Probability.ExponentialRateCertificate (GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore)
    (GGSG19TopThree.approvalPairwiseRate pUp pDown) ∨
  ∀ᶠ (n : ℕ) in Filter.atTop, GGSG19TopThree.pairwiseScoringErrorProb law hiScore loScore n = 0

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proposition3_lem_pairwiselearning_approval_finite_ternary

Recorded source-to-Spec screening **Verdict:** matches

Reason: For the source ternary K-Approval score gap, the target uses the displayed closed rate -log(2 sqrt(pUp pDown) + pZero) and the associated error sequence. Its zero-error branch explicitly covers the boundary that the source formula approaches, while preserving the ordinary strict source regime.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 4

Source locator: cited publication, lines 522-538

Verbatim source input

```
\parbold{Rates for learning the outcome}
In general, the rates at which one learns each pair of candidates immediately translate to rates for learning the entire outcome $O^*$.

\begin{restatable}{proposition}{thmgoallearning}
	\label{thm:goal_learning}
	Consider goal $G$ and $\beta\in\mB$ such that $O^N \to O^*$. Let $Q^N$ be the expected number of errors in the outcome after $N$ voters,
	↪  $\sum_{i\in C_s^*,j\in C_t^*, s < t} \pr(\sigma^N(i)>\sigma^N(j))$. Then $Q^N$ goes to zero with large deviation rate
%
	\[
	r(\beta) = \min_{i\in C_s^*,j\in C_t^*, s < t} r_{ij}(\beta) \label{eqnpart:outcomerate}
	\]
	Further, the following upper bound holds for any $N$. \[Q^N \leq M^2 \exp(-rN)\] %
%
%
%

%
\end{restatable}

[Next verbatim source excerpt]

\thmgoallearning*
\begin{proof}
%
%

By the Union bound
\begin{align*}
%
Q^N &= \sum_{i\in C_s^*,j\in C_t^*, s < t} \pr(\sigma^N(i) > \sigma^N(j)) \\
%
&\leq \sum_{i\in C_s^*,j\in C_t^*, s < t} \exp[-r_{ij}N] & \text{Proposition }~\ref{thm:pairwiselearning}\\
&\leq M^2  \exp[-rN]\\
%
%
%
%
%
%
\end{align*}

%
Now, using large deviation properties:

%
By supposition, $Q^N \to 0$, and so $-Q^N$ approaches $0$ from below.
%
Then,
\begin{align}
%
-\lim_{N\to\infty} \frac{1}{N}\log(Q^N)
&=  -\lim_{N\to\infty} \frac{1}{N}\log \sum_{i\in C_s^*,j\in C_t^*, s < t} \pr(\sigma^N(i) > \sigma^N(j))\nonumber\\
&= -\max_{i\in C_s^*,j\in C_t^*, s < t}\left(\lim_{N\to\infty} \frac{1}{N}\log \pr(\sigma^N(i) > \sigma^N(j))\right)\label{eqnstep:ldpproperty}\\
&= \min_{i\in C_s^*,j\in C_t^*, s < t}-\left(\lim_{N\to\infty} \frac{1}{N}\log \pr(\sigma^N(i) > \sigma^N(j))\right)\nonumber\\
&= \min_{i\in C_s^*,j\in C_t^*, s < t} r_{ij} = r\nonumber
\end{align}

Line~\eqref{eqnstep:ldpproperty} follows from: $\forall a^\epsilon_i \geq 0$, ${\lim \sup}_{\epsilon \to 0} \left[\epsilon\log\left(\sum_i^N
↪  a^\epsilon_i\right)\right] = \max^N_i {\lim \sup}_{\epsilon \to 0} \epsilon \log(a^\epsilon_i)$.
See, e.g., Lemma 1.2.15 in~\cite{dembo_large_2010} for a proof of this property.

Thus $r$ is the large deviation rate for $Q^N$.

\end{proof}
```

Expanded PaperInterface specification

```
∀ {Pair : Type u_1} {Candidate : Type u_2} {Signal : Type u_3} [inst : Fintype Pair] [inst_1 : Nonempty Pair]
  [DecidableEq Pair] [inst_3 : Fintype Signal] [inst_4 : DecidableEq Signal] (law : PMF Signal)
  (score : Candidate → Signal → ℝ) (hi lo : Pair → Candidate),
  (∀ (pair : Pair),
     0 ≤ AppliedModelingLib.pmfExp law fun (signal : Signal) => score (hi pair) signal - score (lo pair) signal) →
    AppliedModelingLib.Probability.HasExtendedExponentialRate
      (fun (sampleSize : ℕ) =>
        ∑ pair : Pair, GGSG19TopThree.finiteScoreGapPairwiseErrorProb law score (hi pair) (lo pair) sampleSize)
      (GGSG19TopThree.finiteOutcomeLearningExtendedRate fun (pair : Pair) =>
        GGSG19TopThree.pairwiseScoringExtendedRate law (score (hi pair)) (score (lo pair)))
```

Retained governing declarations

- AppliedModelingLib.Probability.HasExtendedExponentialRate
- AppliedModelingLib.pmfExp

- GGSG19TopThree.finiteOutcomeLearningExtendedRate
- GGSG19TopThree.finiteScoreGapPairwiseErrorProb
- GGSG19TopThree.pairwiseScoringExtendedRate

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Type u_3
4. parameter: Fintype Pair
5. assumption: Nonempty Pair
6. parameter: DecidableEq Pair
7. parameter: Fintype Signal
8. parameter: DecidableEq Signal
9. parameter: PMF Signal
10. parameter: Candidate → Signal → ℝ
11. parameter: Pair → Candidate
12. parameter: Pair → Candidate
13. assumption: ∀ (pair : Pair),
  0 ≤ AppliedModelingLib.pmfExp law fun (signal : Signal) => score (hi pair) signal - score (lo pair) signal
14. conclusion: AppliedModelingLib.Probability.HasExtendedExponentialRate
  (fun (sampleSize : ℕ) =>
    ∑ pair : Pair, GGSG19TopThree.finiteScoreGapPairwiseErrorProb law score (hi pair) (lo pair) sampleSize)
  (GGSG19TopThree.finiteOutcomeLearningExtendedRate fun (pair : Pair) =>
    GGSG19TopThree.pairwiseScoringExtendedRate law (score (hi pair)) (score (lo pair)))

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proposition4_thm_goal_learning_exact_minimum_rate

Recorded source-to-Spec screening **Verdict:** matches

Reason: The target is the source sum of cross-tier reversal probabilities at the minimum pairwise rate. The extended-rate reading makes the source formula explicit when a pair has no errors after a positive sample.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 5

Source locator: cited publication, lines 522-538

Verbatim source input

```
\parbold{Rates for learning the outcome}
In general, the rates at which one learns each pair of candidates immediately translate to rates for learning the entire outcome $O^*$.

\begin{restatable}{proposition}{thmgoallearning}
	\label{thm:goal_learning}
	Consider goal $G$ and $\beta\in\mB$ such that $O^N \to O^*$. Let $Q^N$ be the expected number of errors in the outcome after $N$ voters,
	↪  $\sum_{i\in C_s^*,j\in C_t^*, s < t} \pr(\sigma^N(i)>\sigma^N(j))$. Then $Q^N$ goes to zero with large deviation rate
%
	\[
	r(\beta) = \min_{i\in C_s^*,j\in C_t^*, s < t} r_{ij}(\beta) \label{eqnpart:outcomerate}
	\]
	Further, the following upper bound holds for any $N$. \[Q^N \leq M^2 \exp(-rN)\] %
%
%
%

%
\end{restatable}

[Next verbatim source excerpt]

\thmgoallearning*
\begin{proof}
%
%

By the Union bound
\begin{align*}
%
Q^N &= \sum_{i\in C_s^*,j\in C_t^*, s < t} \pr(\sigma^N(i) > \sigma^N(j)) \\
%
&\leq \sum_{i\in C_s^*,j\in C_t^*, s < t} \exp[-r_{ij}N] & \text{Proposition }~\ref{thm:pairwiselearning}\\
&\leq M^2  \exp[-rN]\\
%
%
%
%
%
%
\end{align*}

%
Now, using large deviation properties:

%
By supposition, $Q^N \to 0$, and so $-Q^N$ approaches $0$ from below.
%
Then,
\begin{align}
%
-\lim_{N\to\infty} \frac{1}{N}\log(Q^N)
&=  -\lim_{N\to\infty} \frac{1}{N}\log \sum_{i\in C_s^*,j\in C_t^*, s < t} \pr(\sigma^N(i) > \sigma^N(j))\nonumber\\
&= -\max_{i\in C_s^*,j\in C_t^*, s < t}\left(\lim_{N\to\infty} \frac{1}{N}\log \pr(\sigma^N(i) > \sigma^N(j))\right)\label{eqnstep:ldpproperty}\\
&= \min_{i\in C_s^*,j\in C_t^*, s < t}-\left(\lim_{N\to\infty} \frac{1}{N}\log \pr(\sigma^N(i) > \sigma^N(j))\right)\nonumber\\
&= \min_{i\in C_s^*,j\in C_t^*, s < t} r_{ij} = r\nonumber
\end{align}

Line~\eqref{eqnstep:ldpproperty} follows from: $\forall a^\epsilon_i \geq 0$, ${\lim \sup}_{\epsilon \to 0} \left[\epsilon\log\left(\sum_i^N
↪  a^\epsilon_i\right)\right] = \max^N_i {\lim \sup}_{\epsilon \to 0} \epsilon \log(a^\epsilon_i)$.
See, e.g., Lemma 1.2.15 in~\cite{dembo_large_2010} for a proof of this property.

Thus $r$ is the large deviation rate for $Q^N$.

\end{proof}
```

Expanded PaperInterface specification

```
∀ {Pair : Type u_1} {Candidate : Type u_2} {Signal : Type u_3} [inst : Fintype Pair] [inst_1 : Nonempty Pair]
  [inst_2 : Fintype Candidate] [inst_3 : Fintype Signal] [inst_4 : DecidableEq Signal] (law : PMF Signal)
  (score : Candidate → Signal → ℝ) (hi lo : Pair → Candidate),
  (∀ (pair : Pair),
     0 ≤ AppliedModelingLib.pmfExp law fun (signal : Signal) => score (hi pair) signal - score (lo pair) signal) →
    Fintype.card Pair ≤ Fintype.card Candidate ^ 2 →
      ∀ (n : ℕ),
        ∑ pair : Pair, GGSG19TopThree.finiteScoreGapPairwiseErrorProb law score (hi pair) (lo pair) n ≤
          ↑(Fintype.card Candidate) ^ 2 *
            Real.exp
              (-↑n *
                GGSG19TopThree.finiteOutcomeLearningRate fun (pair : Pair) =>
                  GGSG19TopThree.pairwiseScoringRate law (score (hi pair)) (score (lo pair)))
```

Retained governing declarations

- AppliedModelingLib.pmfExp

- GGSG19TopThree.finiteOutcomeLearningRate
- GGSG19TopThree.finiteScoreGapPairwiseErrorProb
- GGSG19TopThree.pairwiseScoringRate

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Type u_3
4. parameter: Fintype Pair
5. assumption: Nonempty Pair
6. parameter: Fintype Candidate
7. parameter: Fintype Signal
8. parameter: DecidableEq Signal
9. parameter: PMF Signal
10. parameter: Candidate → Signal → ℝ
11. parameter: Pair → Candidate
12. parameter: Pair → Candidate
13. assumption: ∀ (pair : Pair),
  0 ≤ AppliedModelingLib.pmfExp law fun (signal : Signal) => score (hi pair) signal - score (lo pair) signal
14. assumption: Fintype.card Pair ≤ Fintype.card Candidate ^ 2
15. parameter: ℕ
16. conclusion: Real.le† (∑ pair : Pair, GGSG19TopThree.finiteScoreGapPairwiseErrorProb law score (hi pair) (lo pair) n)
  (↑(Fintype.card Candidate) ^ 2 *
    Real.exp
      (-↑n *
        GGSG19TopThree.finiteOutcomeLearningRate fun (pair : Pair) =>
          GGSG19TopThree.pairwiseScoringRate law (score (hi pair)) (score (lo pair))))

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proposition4_thm_goal_learning_finite_sample_M_sq_bound

Recorded source-to-Spec screening **Verdict:** matches

Reason: The target is the source every-N union-bound estimate: the same relevant-pair error sum is bounded by the candidate-cardinality squared factor times the exponential at the minimum pairwise rate.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 6

Source locator: cited publication, lines 680-690

Verbatim source input

Our first result is that randomizing between scoring rules does not help, for \textit{any} voter noise model, in contrast to the case when restricted to approval votes.

\begin{restatable}{theorem}{lemrandomizebetterscoring}
\label{lem:randomizebetterscoring}
Randomization does not improve the outcome learning rate for any asymptotically design-invariant noise model F or goal G. For any randomized scoring rule mechanism (B,D), where $B\subset \mB$, for any F, G, the scoring rule $\beta^*(k) = \sum_{p} d_p \beta_p(k)$ satisfies $r(\beta^*) \geq r(B,D)$.

%
%
\end{restatable}

[Next verbatim source excerpt]

\lemrandomizebetterscoring*
\begin{proof}
From Proposition~\ref{thm:pairwiselearning}, for a given scoring rule β and pair of candidates i,j, the learning rate is
\begin{align*}
r_{ij}(\beta) &= -\inf_{z\in\bbR} \log \bbE_F\left[\exp\left[z\left[\beta(\sigma_v(i)) - \beta(\sigma_v(j))\right]\right]\right]\\
&= - \log \inf_{z\in\bbR} \bbE_F\left[\exp\left[z\left[\beta(\sigma_v(i)) - \beta(\sigma_v(j))\right]\right]\right]
\end{align*}

Similarly, if we use scoring rules $\{\beta^u\}_{u=1}^P$, each with probability d^u, then,
\begin{align*}
r_{ij}(\{\beta^u\}_{u=1}^P) &= - \log \inf_{z\in\bbR} \bbE_{F,\{d^u,\beta^u\}}\left[\exp\left[z\left[\beta^u(\sigma_v(i)) - \beta^u(\sigma_v(j))\right]\right]\right]\\
&= - \log \inf_{z\in\bbR} \sum_u d^u \bbE_{F}\left[\exp\left[z\left[\beta^u(\sigma_v(i)) - \beta^u(\sigma_v(j))\right]\right]\right]
\end{align*}

Now, for a single scoring rule $\beta(\cdot)$, let \[\gamma(\beta(1) , \dots, \beta(M)) \triangleq \bbE_F\left[\exp\left[z\left[\beta(\sigma_v(i)) - \beta(\sigma_v(j))\right]\right]\right]\]

Below, we show that $\gamma(\beta(1) , \dots, \beta(M))$ is convex in $\beta(k), \forall k, z$. Then, by convexity, $\forall z$
\begin{align*}
\sum_{u=1}^{P} d^u \gamma(\beta^u(1), \dots, \beta^u(M))& \geq
\gamma\left(\sum_{u=1}^{P} d^u\beta^u(1) , \dots, \sum_{u=1}^{P} d^u\beta^u(M)\right)
%
%
\end{align*}
and so
\begin{align*}
\inf_z\left[\sum_{u=1}^{P} d^u \gamma(\beta^u(1), \dots, \beta^u(M))\right]& \geq
\inf_z \gamma\left(\sum_{u=1}^{P} d^u\beta^u(1) , \dots, \sum_{u=1}^{P} d^u\beta^u(M)\right)
\end{align*}

The left hand side is equal to the argument inside the $-\log(\cdot)$ for the rate function for randomizing between scoring rules $\{\beta^u\}_{u=1}^P$, each with probability d^u, and the right hand side is the argument inside for the rate function for instead using the single scoring rule β^* defined as the convex combination of $\{\beta^u\}_{u=1}^P$. Then, as $-\log(x)$ is decreasing in x, we have that \[r_{ij}(\beta^*) \geq r_{ij}(\{\beta^u\}_{u=1}^P)\].

As this holds for each pair of candidates i,j simultaneously, we are done.

\parbold{Proof that $\gamma(\beta(1) , \dots, \beta(M)) \triangleq \bbE_F\left[\exp\left[z\left[\beta(\sigma_v(i)) - \beta(\sigma_v(j))\right]\right]\right]$ is convex in $\beta(k), \forall k$}

We directly calculate the Hessian of γ and note that it is diagonally dominant and thus positive semidefinite. For notational convenience, we let $\beta_k = \beta(k)$, and $\sigma(k,\ell) = \pr_F(\sigma_v(i) = k, \sigma_v(j) = \ell)$. Of course, $\sigma(k,k)=0$, as we assume each voter has a strict ranking as her preference.
\begin{align*}
\gamma(\beta_1,\dots,\beta_M) &= \bbE_F\left[\exp\left[z\left[\beta(\sigma_v(i)) - \beta(\sigma_v(j))\right]\right]\right]\\
&= \sum_{k=1}^{M}\sum_{\ell = 1}^M \sigma(k,\ell) \exp\left[z \left[\beta_k - \beta_\ell \right]\right]\\
\frac{\partial}{\partial \beta_k} \gamma(\beta_1,\dots,\beta_M) &= z\exp\left[z \beta_k\right]\sum_{\ell\neq k}\exp\left[-z \beta_\ell\right]\sigma(k,\ell) - z\exp\left[-z \beta_k\right]\sum_{\ell\neq k}\exp\left[z \beta_\ell\right]\sigma(\ell,k)\\
\frac{\partial^2}{\partial \left(\beta_k\right)^2} \gamma(\beta_1,\dots,\beta_M) &= z^2\exp\left[z \beta_k\right]\sum_{\ell\neq k}\exp\left[-z \beta_\ell\right]\sigma(k,\ell) + z^2\exp\left[-z \beta_k\right]\sum_{\ell\neq k}\exp\left[z \beta_\ell\right]\sigma(\ell,k)\\
&= z^2 \left[\left[\exp\left[z \beta_k\right]\sum_{\ell\neq k}\exp\left[-z \beta_\ell\right]\sigma(k,\ell)\right] + \left[\exp\left[-z \beta_k\right]\sum_{\ell\neq k}\exp\left[z \beta_\ell\right]\sigma(\ell,k)\right]\right]\\
\frac{\partial^2}{\partial \beta_k\beta_\ell}\gamma(\beta_1,\dots,\beta_M)&= -z^2\left[\exp\left[z \beta_k\right]\exp\left[-z \beta_\ell\right]\sigma(k,\ell) + \exp\left[-z \beta_k\right]\exp\left[z \beta_\ell\right]\sigma(\ell,k)\right]
\end{align*}
Thus, the Hessian of γ is diagonally dominant with non-negative diagonal elements: $\forall k$,
$$ \left|\frac{\partial^2 \gamma}{\partial \left(\beta_k\right)^2} \right| \geq \sum_{\ell\neq k}\left|\frac{\partial^2 \gamma}{\partial \beta_k\beta_\ell} \right|$$
and so the Hessian is positive semi-definite. Thus, γ is convex in β_k.

%
%
%
%

%
%
%

Recorded source-to-Spec screening **Verdict:** matches
Reason: The target formalizes Theorem 1's rate core: a finite randomized reasonable-scoring mixture has outcome rate no larger than its probability-weighted static score-gap combination. The source design-invariant goal and reasonable-rule context is carried by the typed model/prerequisite layer while the target exposes the weighted finite rate inequality itself.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 7

Source locator: cited publication, lines 702-722

Verbatim source input

Next, we further refine the result of~\citet{caragiannis_learning_2017}, by showing that the ``pivotal pair'' feature of their example -- where different pairs of candidates dominate the learning rate for different mechanisms -- is key. In particular, our next result establishes, again for any noise model, that randomization amongst K-Approval mechanisms cannot help separate any \textit{given} pair of candidates.

%

\begin{restatable}{theorem}{lemrandomizenotbetterapproval}
\label{lem:randomizenotbetterapproval}

Randomization amongst K-Approval mechanisms does not improve the learning rate for separating a given pair of candidates i,j for any asymptotically design-invariant noise model F or goal G. For any randomized K-Approval mechanism (B,D), where $\beta_p\in B$ corresponds to p-Approval, for any F, G, there exists a mechanism K_{ij}^*-Approval such that $r_{ij}(K_{ij}^*) \geq r_{ij}(B,D)$.

%

%

%
%
%
\end{restatable}

[Next verbatim source excerpt]

\lemrandomizenotbetterapproval*
\begin{proof}
From Proposition~\ref{lem:pairwiselearning_approval}, for k-Approval,
\begin{align*}
r_{ij}(t^i_{ij}, t^j_{ij}) &= -\log\left[2\sqrt{t_{ij}^it_{ij}^j} + 1 - t_{ij}^i - t_{ij}^j \right]
\end{align*}
Where t^i_{ij} is the probability that i is approved but j is not.

This rate function is convex in t^i_{ij}, t^j_{ij}:
\begin{itemize}
\item $r_{ij}(a,b) = h(g(a, b))$, where $h(x) = -\log(x)$, $g(a, b) = 2\sqrt{ab} + 1 - a - b$.
\item $g(a, b)$ is concave in a, b
\item h is convex, and $\tilde{h}$ is non-increasing, where $\tilde{h}(x) = \begin{cases} h(x) & x > 0 \\ \infty & x \leq 0 \end{cases}$ is the extended value function of h.
\item By convex composition rules, $r_{ij}(a,b)$ is convex (see, e.g., page 84 of~\citet{boyd_convex_2004}).
\end{itemize}

The result follows by convexity. Consider a randomization of K-Approval mechanisms for $K\in\{1, \dots, M-1\}$, where K-Approval is used with probability d^K.

%

The resulting approval probabilities are: $t^i_{ij} = \sum_{K=1}^{M-1} d^K t^i_{ij}(K), t^j_{ij} = \sum_{K=1}^{M-1} d^K t^j_{ij}(K)$. By convexity:
\begin{align*}
r\left(\sum_{K=1}^{M-1} d^K t^i_{ij}(K), \sum_{K=1}^{M-1} d^K t^j_{ij}(K)\right) &\leq \sum_{K=1}^{M-1} d^K r\left(t^i_{ij}(K), t^j_{ij}(K)\right)\\
&= \sum_{K=1}^{M-1} d^K r_{ij}(K)\\
&\leq \max_K r_{ij}(K)
\end{align*}

We note that, unlike the previous proof, we cannot conclude in general that randomization cannot improve the rates at which the outcome is learned (in fact, Theorem~\ref{lem:randomizebetterapproval_Wselection} establishes otherwise). That is because while the same β^* could be said to be rate optimal (compared to the randomized mechanism) for every pair of candidates simultaneously in that proof, in this proof $\arg\max_K r_{ij}(K)$ may change based on the pair i,j.

\end{proof}

Expanded PaperInterface specification

```
∀ {n : ℕ} {Rule : Type u_1} [inst : Fintype Rule] [Nonempty Rule]
  (law : PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)) (K : Rule → ℕ) (weight : Rule → ℝ),
  (∀ (rule : Rule), 0 ≤ weight rule) →
    ∑ rule : Rule, weight rule = 1 →
      ∀ (hi lo : AppliedModelingLib.SocialChoice.Ranking.Candidate n),
        ∃ (rule : Rule),
          ↑(GGSG19TopThree.approvalPairwiseRate
              (∑ rule : Rule,
                weight rule * AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb law (K rule) hi lo)
              (∑ rule : Rule,
                weight rule * AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb law (K rule) hi lo)) ≤
            GGSG19TopThree.approvalPairwiseExtendedRate
              (AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb law (K rule) hi lo)
              (AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb law (K rule) hi lo)
```

Retained governing declarations

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb
- AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb
- GGSG19TopThree.approvalPairwiseExtendedRate
- GGSG19TopThree.approvalPairwiseRate

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: Type u_1
3. parameter: Fintype Rule
4. assumption: Nonempty Rule
5. parameter: PMF (AppliedModelingLib.SocialChoice.Ranking.Ranking n)
6. parameter: Rule → ℕ
7. parameter: Rule → ℝ
8. assumption: ∀ (rule : Rule), 0 ≤ weight rule
9. assumption: ∑ rule : Rule, weight rule = 1
10. parameter: AppliedModelingLib.SocialChoice.Ranking.Candidate n
11. parameter: AppliedModelingLib.SocialChoice.Ranking.Candidate n
12. conclusion: ∃ (rule : Rule),
  ↑(GGSG19TopThree.approvalPairwiseRate
      (∑ rule : Rule, weight rule * AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb law (K rule) hi lo)
      (∑ rule : Rule,
        weight rule * AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb law (K rule) hi lo)) ≤
    GGSG19TopThree.approvalPairwiseExtendedRate
      (AppliedModelingLib.SocialChoice.Ranking.kApprovalPairUpProb law (K rule) hi lo)
      (AppliedModelingLib.SocialChoice.Ranking.kApprovalPairDownProb law (K rule) hi lo)

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_theorem2_lem_randomizenotbetterapproval_pairwise

Recorded source-to-Spec screening **Verdict:** matches

Reason: For every fixed pair, the target selects one static K-Approval component whose extended pairwise rate weakly dominates the randomized mixture rate. This is the source theorem's fixed-pair non-improvement conclusion with explicit finite mixture weights.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 8

Source locator: cited publication, lines 726-734

Verbatim source input

This theorem directly implies that, for the Mallows model, randomization among K-Approval voting cannot speed up learning when the goal is to identify a set of W winners, as opposed to when the goal is to rank.

\begin{restatable}{corollary}{lemmallowsnorando}
\label{lem:mallowsnorando}
Randomization among K-Approval mechanisms does not improve the learning rate for selecting W winners from the Mallows model.
For any randomized K-Approval mechanism (B,D), where $\beta_p\in B$ corresponds to p-Approval, for selecting W winners from the Mallows model, there exists an Approval rate optimal mechanism K^*-Approval such that $r(K^*) \geq r(B,D)$.

\end{restatable}

The proof simply notes that under the Mallows model with this goal, the candidate pair $W, W+1$ (when candidates are indexed according to reference distribution σ^*) is pivotal regardless of the K-Approval mechanism used.

[Next verbatim source excerpt]

\lemmallowsnorando*
\begin{proof}
When selecting W winners out of M candidates, we need to separate candidates $1 \dots W$ from candidates $W + 1 \dots M$. It is easy to show that the pivotal pair, regardless of which K is used in K-Approval, is W,$W+1$. Applying Theorem~\ref{lem:randomizenotbetterapproval}, then, randomization cannot help the overall rate.
\end{proof}

Expanded PaperInterface specification

```
∀ {n : ℕ} (center : AppliedModelingLib.SocialChoice.Ranking.Ranking n)
  (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n),
  0 < ↑W →
    ∀ (randomizedRate : WithTop ℝ),
      AppliedModelingLib.Probability.HasExtendedExponentialRate
        (GGSG19TopThree.ProofBridge.zeroNoiseTopWApprovalError center W) ⊤ ∧
       randomizedRate ≤ ⊤
```

Retained governing declarations

- AppliedModelingLib.Probability.HasExtendedExponentialRate
- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.ProofBridge.zeroNoiseTopWApprovalError

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: AppliedModelingLib.SocialChoice.Ranking.Ranking n
3. parameter: AppliedModelingLib.SocialChoice.Ranking.Candidate n
4. assumption: 0 < ↑W
5. parameter: WithTop ℝ
6. conclusion: AppliedModelingLib.Probability.HasExtendedExponentialRate
   (GGSG19TopThree.ProofBridge.zeroNoiseTopWApprovalError center W) ⊤ ∧
  randomizedRate ≤ ⊤

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorando_zero_noise

Recorded source-to-Spec screening **Verdict:** matches

Reason: At the source zero-noise endpoint, the selected static W-Approval error is eventually zero and hence has top extended rate. The target states that fact and the resulting weak dominance over every randomized extended rate.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 9

Source locator: cited publication, lines 726-734

Verbatim source input

This theorem directly implies that, for the Mallows model, randomization among K-Approval voting cannot speed up learning when the goal is to identify a set of W winners, as opposed to when the goal is to rank.

\begin{restatable}{corollary}{lemmallowsnorando}
\label{lem:mallowsnorando}
Randomization among K-Approval mechanisms does not improve the learning rate for selecting W winners from the Mallows model. For any randomized K-Approval mechanism (B,D), where $\beta_p\in B$ corresponds to p-Approval, for selecting W winners from the Mallows model, there exists an Approval rate optimal mechanism K^*-Approval such that $r(K^*) \geq r(B,D)$.

\end{restatable}

The proof simply notes that under the Mallows model with this goal, the candidate pair $W, W+1$ (when candidates are indexed according to reference distribution σ^*) is pivotal regardless of the K-Approval mechanism used.

[Next verbatim source excerpt]

\lemmallowsnorando*
\begin{proof}
When selecting W winners out of M candidates, we need to separate candidates $1 \dots W$ from candidates $W + 1 \dots M$. It is easy to show that the pivotal pair, regardless of which K is used in K-Approval, is W,$W+1$. Applying Theorem~\ref{lem:randomizenotbetterapproval}, then, randomization cannot help the overall rate.
\end{proof}

Expanded PaperInterface specification

```
∀ {n : ℕ} (center : AppliedModelingLib.SocialChoice.Ranking.Ranking n) (q : ℝ)
  (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n)
  (hDomain : GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain q W) {Rule : Type u_1}
  [inst : Fintype Rule] [Nonempty Rule] (K : Rule → ℕ),
  (∀ (rule : Rule), 0 < K rule) →
    (∀ (rule : Rule), K rule < n + 2) →
      ∀ (weight : Rule → ℝ),
        (∀ (rule : Rule), 0 ≤ weight rule) →
          ∑ rule : Rule, weight rule = 1 →
            ∃ (cut : GGSG19TopThree.RankingProperPrefixCut n),
              (∀ (cut' : GGSG19TopThree.RankingProperPrefixCut n),
                 GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
                   (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
                     (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
                   W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain)
                   (↑cut' + 1) ≤
                 GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
                   (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
                     (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
                   W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain)
                   (↑cut + 1)) ∧
               GGSG19TopThree.mallowsTopWRandomizedKApprovalOutcomeRate
                 (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
                   (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
                 W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) K
                 weight ≤
               GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
                 (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
                   (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
                 W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain)
                 (↑cut + 1)
```

Retained governing declarations

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.RankingProperPrefixCut
- GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain
- GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
- GGSG19TopThree.mallowsTopWRandomizedKApprovalOutcomeRate

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: AppliedModelingLib.SocialChoice.Ranking.Ranking n
3. parameter: ℝ
4. parameter: AppliedModelingLib.SocialChoice.Ranking.Candidate n
5. assumption: GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain q W

```
6. parameter: Type u_1
7. parameter: Fintype Rule
8. assumption: Nonempty Rule
9. parameter: Rule → ℕ
10. assumption: ∀ (rule : Rule), 0 < K rule
11. assumption: ∀ (rule : Rule), K rule < n + 2
12. parameter: Rule → ℝ
13. assumption: ∀ (rule : Rule), 0 ≤ weight rule
14. assumption: ∑ rule : Rule, weight rule = 1
15. conclusion: ∃ (cut : GGSG19TopThree.RankingProperPrefixCut n),
  (∀ (cut' : GGSG19TopThree.RankingProperPrefixCut n),
     GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
        (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
          (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
        W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) (↑cut' + 1) ≤
      GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
        (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
          (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
        W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) (↑cut + 1)) ∧
   GGSG19TopThree.mallowsTopWRandomizedKApprovalOutcomeRate
      (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
        (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
      W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) K weight ≤
     GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
      (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
        (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
      W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) (↑cut + 1)
```

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorando

Recorded source-to-Spec screening **Verdict:** matches

Reason: For the source positive-noise Mallows W-winner setting, the target supplies a static approval cutoff that maximizes the outcome rate and whose rate weakly dominates every finite randomized K-Approval mixture. The nontrivial Mallows and cutoff domains are explicit source-model prerequisites.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 10

Source locator: cited publication, lines 739-748

Verbatim source input

This corollary does not extend to arbitrary noise models, where randomization amongst K-approval mechanisms may improve the learning rate.

\begin{restatable}{theorem}{lemrandomizebetterapprovalWselection}
\label{lem:randomizebetterapproval_Wselection}
Randomization among K-Approval mechanisms may improve the learning rate for the goal of selecting W winners. There exist asymptotically design-invariant settings (M,F) for the goal of selecting W winners such that a randomized K-Approval mechanism (B,D), where $\beta_p\in B$ corresponds to p-Approval, satisfies \[r(B,D) > \max_K r(K) \]
%
%
\end{restatable}

We prove the result two ways: (1) we construct an example in which candidate h is asymptotically selected, and candidates i,j are not. Which of $h \succ i$ or $h \succ j$ is the pivotal pair (determines the overall rate function) depends on the K-Approval mechanism used, and randomizing between two mechanisms improves the overall rate; (2) perhaps more interestingly, we find many examples in our real PB elections and other ranking data in which randomization would have sped up learning for the task of selecting a set of winning candidates (see Section~\ref{sec:randomizationpractice}).

[Next verbatim source excerpt]

\lemrandomizebetterapprovalWselection*
\begin{proof}
We provide two proofs: a numeric example from a real-world election, and a contrived, constructed example.

\parbold{Numeric example found in a real election} In Durham Ward 1, to select 2 winners, randomizing between 3 and 4-Approval is better than either individually, even though asymptotically the mechanisms pick the same set of winners. The critical pair with 2-Approval is with the candidate asymptotically ranked 1st, and the best item not selected. With 3-Approval, it is with the candidate asymptotically ranked 2nd, and the same best item not selected.

We will call these items h, i,j (the one not selected) respectively. The respective probabilities of being selected alone:

%
%
%
%
%
%
%
%
%
%
%

\begin{align*}
t_{hj}^h(3)&= 0.277 \\
t_{hj}^h(4)&= 0.266 \\
t_{hj}^j(3)&= 0.200 \\
t_{hj}^j(4)&= 0.188 \\
t_{ij}^i(3)&= 0.255 \\
t_{ij}^i(4)&= 0.295 \\
t_{ij}^j(3)&= 0.160\\
t_{ij}^j(4)&= 0.217\\\\
%
t_{hj}^h(\{3, 4\})&= 0.271 \\
t_{hj}^j(\{3, 4\})&= 0.194 \\
t_{ij}^i(\{3, 4\})&= 0.275 \\
t_{ij}^j(\{3, 4\})&= 0.189\\
\end{align*}

And the resulting rates (using the formula in Proposition~\ref{lem:pairwiselearning_approval}) are:

\begin{align*}
r_{hj}(3) &= .00616932\\
r_{ij}(3) &= .01114061\\
r_{hj}(4) &= .00677352\\
r_{ij}(4) &= .00592327\\
r_{hj}(\{3, 4\}) &= .00642839\\
r_{ij}(\{3, 4\}) &= .00815633\\
r(3) &= \min(r_{hj}(3), r_{ij}(3)) = .00616932\\
r(4) &= \min(r_{hj}(4), r_{ij}(4)) = .00592327\\
r(\{3, 4\}) &= \min(r_{hj}(\{3, 4\}), r_{ij}(\{3, 4\})) = .00642839 > \max(r(3), r(4))
\end{align*}

Thus randomization improves learning.

%
%
%
%
%
%
%
%
%
%
%

\parbold{Constructed example with design invariance}
We now construct a fully design-invariant example with the same flavor as the numeric example, where which pair is critical changes with the mechanism.

Consider three candidates h,i,j, such that h is asymptotically in the set of W winners and i,j are not. Thus, we need the rates at which h is separated from both i,j. Let $W=K < L=K+1$.

We prove the result by giving an example where: it is easier to separate h from i using K-Approval, and easier to separate h from j using L-Approval. Using K-Approval, r_{hj} asymptotically dominates the rate at which the overall outcome is learned, and using L-Approval, r_{hi} does. Further, randomizing between the two mechanisms improves the two rates that dominate enough such that the overall rate is improved.

We need to show the following hold for our example: one of the rates between candidates h and i,j are smaller than other rates, i.e., dominate the overall learning rate when K and/or L approval is used; randomization between K and L approval helps the minimum rate between candidates h and i,j; K'-Approval ($K'\neq K, K' \neq L$) produces a worse rate than either K or L approval; and this example is asymptotically design-invariant.

We prove each of these conditions in turn after specifying the example.

Recall that $t_{ij}^i(k)$ is the probability that i is approved but j is not, using k-Approval. Here, we will use: $$t_{hi}^i(K),t_{hi}^i(L),t_{hi}^h(K),t_{hi}^h(L), t_{hj}^j(K),t_{hj}^j(L),t_{hj}^h(K),t_{hj}^h(L)$$. The end row labeled ``Total value'' then sums up these values.

%

%

\parbold{Example Specification}
Consider F such according to the following table, where the first column is the probabilities of the positions in the second set of columns. The third set of columns indicates whether those set of positions contribute to the given probabilities, for easy accounting.
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
\FloatBarrier
\begin{table}[h]
\hspace*{-1cm}
\begin{tabular}{c|c|ccc||cccccccc}
%
& & \multicolumn{3}{c||}{Positions of h,i,j} & \multicolumn{8}{c}{Contributes to? (Y = Yes)}\\
Row& $Pr_F(\cdot)$ & $\sigma(h)$ & $\sigma(i)$ & $\sigma(j)$ &$t_{hi}^h(K)$ & $t_{hi}^i(K)$ & $t_{hi}^h(L)$ & $t_{hi}^i(L)$ & $t_{hj}^h(K)$ & $t_{hj}^j(K)$ & $t_{hj}^h(L)$ & $t_{hj}^j(L)$\\ \hline
1& $a $ & K & $K+1$ & L &Y&Y&Y&Y&&\\ %
2& a & $K-1$ & K & L &&&&&Y&&&\\
3& $T_1 - {a} - \epsilon$ & K & $K+1$ & $L+2$ &Y&Y&Y&Y&Y&\\
4& T_2-2a & $L+1$ & $L+2$ & K &&&&&Y&Y\\
5& T_2-2a & $L+1$ & K & $L+2$ &&Y&Y&Y&&&\\
6& a & $L+1$ & {K} & L &&Y&Y&Y&&&{Y}\\
7& a & L & $K-1$ & {K} &&Y&&&Y&&\\
8& a & L & $L+1$ & $L+2$ &&&Y&&&&{Y}&\\ %
9& a & $L+1$ & $L+2$ & {L} &&&&&&&Y\\
10& ϵ & \multicolumn{3}{c||}{See caption} &Y&Y&Y&Y&&{Y}&\\
%
11& 0 & \multicolumn{3}{c||}{Otherwise}&&&&&&&\\ \hline
\multicolumn{5}{r}{\textbf{Total value:}}&T_1&T_2&T_1+a&T_2-a&T_1+a&T_2-a&T_1&T_2
\end{tabular} %
\caption{Where the constants such that $0<\epsilon<a<\frac{T_2}{2}<T_2 < T_1<T_1 + 2T_2 + a=1$, i.e., the table describes a valid probability distribution.%
Row 10 is as follows: The first K candidates (the asymptotic winners) occupy the first K spots, in an order drawn uniformly at random. Similarly, the bottom $M-K$ candidates occupy the bottom $M-K$ spots, in an order drawn uniformly at random. This randomization ensures asymptotically design invariance.
}
\end{table}
\FloatBarrier

%

The table does not specify the probabilities of other candidates appearing in any position, so it is possible that they dominate the learning rate (are hardest to learn). (In particular, if the same, asymptotically non-winning candidate q is always in position L in the case in row 3, then it may be hard to separate it from candidate h using L approval). However, we can specify the example further to ensure this does not happen.

%
%

Suppose candidates are indexed by their order in some strict ranking σ^*. Then, candidates $h=K, i = L=K+1, j = K+2$. Further suppose that candidates in $\{1 \dots K-1 \}$ always occupy, in order except in case of row 10, the best positions in a voter's ranking that are not reserved for candidates h,i,j in the table above. %

%

For candidates $q\in\{K+3 \dots M \}$, we have to be more careful to avoid the case in parenthesis above. Suppose these $Q = M-K+2$ candidates fill up the bottom spots in a voter's ranking in a uniform at random order. In other words, they occupy spots $L+3 \dots M$, and the worst spot among whichever of $K, L, L+1,L+2$ is missing in each row in the table above.

\parbold{Rates between the h and i,j dominate the overall learning rate using K or L approval}

We are now ready to show the first claim that learning between candidates h and i,j is hardest (when using either K or L approval).

By the specification above, candidates in $\{1 \dots K-2 \}$ are always approved, and so learning between those candidates and any non-winning candidate is faster than any large deviations rate. Similarly, candidate $K-1$ always is ranked higher than candidates $q\in\{K+3 \dots M \}$, and it is approved alone with high enough probability. %

%

%
%

Then, the other candidates who may dominate the learning rate are candidate $K-1$ (in separation from i, j), or $q\in\{K+3 \dots M \}$ (in separation from h). From the above table:
\begin{align*}
t_{hq}^{h}(K) &= T_1+a & \text{Rows 1,2,3,10}\\
t_{hq}^{q}(K) &= \frac{2a}{Q} & \text{Rows 8,9} \\
t_{hq}^{h}(L) &= \frac{Q-1}{Q}\left[T_1 - \epsilon \right] + 3a + \epsilon & \text{Rows 1,2,7,10; and 3,8 w.p. } \frac{Q-1}{Q} \\
t_{hq}^{q}(L) &= \frac{2T_2 - 3a}{Q} & \text{Rows 4,5,9 w.p.} \frac{1}{Q}\\
%
%
%
t_{(K-1)i}^{K-1}(K) &= T_1+T_2 & \text{Rows } 1,3,4,8,9,10\\%\text{Rows } 1,3,8,9 \text{Rows 1,3,4,5,6,10} \\
t_{(K-1)i}^{i}(K) &= 2a & \text{Rows } 2,7\\
t_{(K-1)j}^{K-1}(K) &= T_1 + T_2+a & \text{Rows } 1,3,5,6,8,9,10\\
t_{(K-1)j}^{j}(K) &= a & \text{Rows } 7\\
t_{(K-1)i}^{K-1}(L) &= T_1+T_2 & \text{Rows } 1,3,4,8,9,10\\
t_{(K-1)i}^{i}(L) &= 2a & \text{Rows } 2,7\\
t_{(K-1)j}^{K-1}(L) &= T_1 + T_2 - 2a & \text{Rows } 3,5,8,10\\
t_{(K-1)j}^{j}(L) &= 2a & \text{Rows } 2,7
\end{align*}
Now, suppose $3a > \frac{T_1}{Q}$ and $Q>2$. (Both conditions occur for Q large enough). Then, applying Remark~\ref{rem:tminusaplusa} regarding learning rates being larger when the arguments are farther away from one another (holding one fixed), the resulting rates with these candidates are dominated by (larger than) the rates between candidates h and i,j, discussed next.

%

%

\parbold{Randomizing improves the minimum rate between candidates h and i,j}
By Remark~\ref{rem:tminusaplusa},
\begin{align*}
r(T_1+a,T_2-a) >r(T_1,T_2)
%
%
\end{align*}

Using K-Approval:
\begin{align*}
\text{Rate between h,i: }\,\,\,

$L=K+1$ & 0 & a & T_2 & $<T_2+\epsilon$ & <1 & <1 & <1 \\
$K+2$ & 0 & 0 & T_2-a & $<T_2+3a+\epsilon$ & <1 & <1 & <1 \\
$q\in\{K+3 \dots M\}$ & 0 & 0 & 0 & $\frac{2a}{Q}$ & $\frac{T_1 + 2T_2 -3a -\epsilon}{Q}$ & <1 & <1 & <1
\end{tabular}
\end{table}
\FloatBarrier
\parbold{Conditions on constants in problem}
For the above claims to hold, we set conditions on the constants in the problem. They are
\begin{align*}
0<\epsilon<a<\frac{T_2}{2}<T_2 &< T_1<T_1 + 2T_2 + a=1\\
%
1 - \epsilon &>2\sqrt{T_1T_2} + 1 - T_1 - T_2\\
\frac{T_1 + 2T_2 -3a -\epsilon}{Q} &< T_1 + 3a\\
3a &> \frac{T_1}{Q}\\
Q&>2\\
1 - T_2 + a - \epsilon &> 2\sqrt{T_1T_2} + 1 - T_1 - T_2
\end{align*}

This is a feasible set of constraints: Q can be set large enough to meet conditions 3,4,5 for any fixed T_1, a, T_2 that meet condition 1. Condition 2 is weaker than the last condition. That leaves the last condition along with the first one.
\begin{align*}
1 - T_2 + a - \epsilon &> 2\sqrt{T_1T_2} + 1 - T_1 - T_2\\
\iff a - \epsilon &> 2\sqrt{T_1T_2} - T_1 \\
\iff \frac{T_2}{2} - 2\epsilon &> 2\sqrt{T_1T_2} - T_1 & \text{set } \frac{T_2}{2}-\epsilon = a \\
\iff \frac{T_2}{2} + T_1 &> 2\epsilon + 2\sqrt{T_1T_2}
\end{align*}
which holds for T_1 large enough, and T_2,ϵ small enough.

\end{proof}

Expanded PaperInterface specification

```
GGSG19TopThree.StrictTopPrefixDominanceOn GGSG19TopThree.constructedWSelectionRanking1TopPrefixProb
    GGSG19TopThree.constructedWSelectionCrossTier ∧
  (Finset.univ.sup'
      GGSG19TopThree.PaperInterface.source_theorem_lem_randomizebetterapproval_w_selection_constructedSpec._proof_1
      fun (K : Fin 4) =>
      GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.constructedWSelectionRanking1StaticKRate K)) <
    GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalRate
```

Retained governing declarations

- GGSG19TopThree.ConstructedWSelectionCandidate
- GGSG19TopThree.ConstructedWSelectionPair
- GGSG19TopThree.StrictTopPrefixDominanceOn
- GGSG19TopThree.constructedWSelectionCrossTier
- GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalRate
- GGSG19TopThree.constructedWSelectionRanking1StaticKRate
- GGSG19TopThree.constructedWSelectionRanking1TopPrefixProb
- GGSG19TopThree.finiteOutcomeLearningRate

Lean-elaborated claim roles

1. conclusion: GGSG19TopThree.StrictTopPrefixDominanceOn GGSG19TopThree.constructedWSelectionRanking1TopPrefixProb
    GGSG19TopThree.constructedWSelectionCrossTier ∧
  (Finset.univ.sup'
      GGSG19TopThree.PaperInterface.source_theorem_lem_randomizebetterapproval_w_selection_constructedSpec._proof_1
      fun (K : Fin 4) =>
      GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.constructedWSelectionRanking1StaticKRate K)) <
    GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.constructedWSelectionRanking1RandomizedApprovalRate

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_theorem_lem_randomizebetterapproval_w_selection_constructed

Recorded source-to-Spec screening **Verdict:** matches

Reason: The target packages the source constructed asymptotically design-invariant W-selection witness: strict top-prefix dominance supplies the selected outcome and the randomized K-Approval rate is strictly larger than the maximum static rate. The separate Durham calculation remains an additional empirical illustration, not a substitute for this existential theorem.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 11

Source locator: cited publication, lines 751-761

Verbatim source input

```
\subsection{$K$-Approval for selecting $W$ winners}
\label{sec:approvaltheory}
One of the most common voting settings is identifying a set of $W$ winners using $K$-Approval, whether in representative democracy elections (typically $K=W=1$), polling for such elections (where the goal often is to identify the top few candidates out of many, especially in primary races), or crowd-sourcing labels (where one wants one or a few labels for an item out of many possible ones). Here, we study how to design such elections, i.e., how to choose the best $K$, i.e., the one that maximizes the learning rate. For simplicity, we work with the Mallows model, extending the resulting insights to real-world data in the next section.


Recall that in a Mallows model, each voter's ranking is a noisy sample from a reference distribution $\sigma^*$. With this symmetric model, one may believe that setting $K=W$ is always optimal. For example, when noise parameter $\phi = 0$ and so each voter's ranking is exactly $\sigma^*$, $K=W$ is optimal; in fact, any other design $K \neq W$ fails to correctly identify the set of winners even asymptotically: it would not distinguish among the first $K$ candidates in $\sigma^*$ or among the last $M-K$ candidates. However, our next result establishes that the cases with $\phi > 0$ are different.

\begin{restatable}{theorem}{lemmallowsnotWK}
    \label{lem:mallowsnotWK}
    Under the Mallows model and the goal of selecting $W$ winners, $W$-Approval may not be Approval rate optimal.
\end{restatable}

[Next verbatim source excerpt]

\lemmallowsnotWK*
\begin{proof}

    We prove the result by providing an example where it is not optimal. Suppose there are 4 candidates, and we wish to select $3$ winners, i.e., separate the first three items from the last item.


Let the items in the reference ranking be, in order, $i=1, 2, 3, 4$, respectively.

A Mallows model (with parameter $\phi = \frac{p}{1-p}$, where $p$ is the probability of flipping a given pair of candidates) can be sampled by repeated insertion \citep{lu14a,diaconis_group_nodate}: starting from the first item in the reference ranking, there exists probability $\tilde{p}_{ij} = \frac{\phi^{i-j}}{1 + \phi^1 + \dots + \phi^{i-1}}$ at which item $i$ can be inserted into position $j\leq i$, \textit{independently} of how items above it were inserted, such that the resulting ranking distribution matches the Mallows model.

Using this repeated insertion property for our example, we can derive $p_{\ell k}$, the probability at which item $3$ is in position $\ell$ and item $4$ is in position $k$ after sampling from a Mallows model with parameter $\phi$.

In particular, if $\ell < k$, $p_{\ell k}$ is exactly the probability that item $3$ is inserted in position $\ell$ and item $4$ is inserted in position $k$. If $k>\ell$, however, it is the probability that item $3$ is inserted in position $\ell - 1$ and then pushed down when item $4$ is inserted in position $k$. (More generally, it turns out, the exact probability for an item appearing in a given position in the Mallows model can be calculated using a simple dynamic program, a fact that does not appear to be documented  elsewhere but may be independently useful. We used this dynamic program to find this given example).

Then, for our example
\begin{align*}
N_i &\triangleq 1 + \phi^1 + \dots + \phi^{i-1} \\
p_{\ell k} &= \begin{bmatrix}
0 & \frac{\phi^2}{N_3}\frac{\phi^2}{N_4} & \frac{\phi^2}{N_3}\frac{\phi^1}{N_4} & \frac{\phi^2}{N_3}\frac{\phi^0}{N_4}\\
\frac{\phi^2}{N_3}\frac{\phi^3}{N_4} & 0 & \frac{\phi^1}{N_3}\frac{\phi^1}{N_4} & \frac{\phi^1}{N_3}\frac{\phi^0}{N_4}\\
\frac{\phi^1}{N_3}\frac{\phi^3}{N_4} & \frac{\phi^1}{N_3}\frac{\phi^2}{N_4} & 0 & \frac{\phi^0}{N_3}\frac{\phi^0}{N_4}\\
\frac{\phi^0}{N_3}\frac{\phi^3}{N_4} & \frac{\phi^0}{N_3}\frac{\phi^2}{N_4} & \frac{\phi^0}{N_3}\frac{\phi^1}{N_4} & 0\\
\end{bmatrix}_{\ell k}
=
\frac{1}{N_3N_4}\begin{bmatrix}
0 & \phi^4 & \phi^3 & {\phi^2}\\
\phi^5 & 0 & \phi^2 & {\phi^1}\\
\phi^4 & \phi^3 & 0 & 1\\
{\phi^3} & {\phi^2} & {\phi^1} & 0\\
\end{bmatrix}_{\ell k}
\end{align*}

Recall that $t_{ij}^i(K)$ is the probability that $i$ is approved but $j$ is not, using $K$-Approval. Then, if we use $3$-approval and $2$-approval, respectively:
\begin{align*}
N_3 &= 1 + \phi + \phi^2\\
N_4 &= 1 + \phi + \phi^2+ \phi^3\\
t_{34}^3(3) &= p_{14} + p_{24} + p_{34}= \frac{\phi^2  + \phi^1 + 1}{N_3N_4}= \frac{1}{N_4}\\
t_{34}^4(3) &= p_{41} + p_{42} + p_{43} = \frac{\phi^3  + \phi^2 + \phi^1}{N_3N_4} = \frac{\phi}{N_4}\\
t_{34}^3(2) &= p_{14} + p_{24} + p_{13} + p_{23} = \frac{\phi^2  + \phi^1 + \phi^3 + \phi^2}{N_3N_4}\\
t_{34}^4(2) &= p_{41} + p_{42} + p_{31} + p_{32} = \frac{\phi^3  + \phi^2 + \phi^4 + \phi^3}{N_3N_4}\\
\end{align*}

Then, recall the rate between items $i,j$ using $K$ approval is
\begin{align*}
    r_{ij}(K) &= -\log\left[ 2\sqrt{t_{ij}^i(K)t_{ij}^j(K)} + 1 - t_{ij}^i(K) - t_{ij}^j(K) \right]\\
     r_{34}(3) &=     -\log\left[ 2\sqrt{\frac{1}{N_4}\frac{\phi}{N_4}} + 1 - \frac{1}{N_4} - \frac{\phi}{N_4} \right]\\
%
     r_{34}(2) &=     -\log\left[ 2\sqrt{\frac{\phi^2  + \phi^1 + \phi^3 + \phi^2}{N_3N_4}\frac{\phi^3  + \phi^2 + \phi^4 + \phi^3}{N_3N_4}} + 1 - \frac{\phi^2  + \phi^1 + \phi^3 + \phi^2}{N_3N_4} - \frac{\phi^3  + \phi^2 + \phi^4 + \phi^3}{N_3N_4} \right]
\end{align*}

When there is low noise, e.g., $\phi = .1$ ($p = .091$):
\begin{align*}
r_{34}(3) &= .5462\\
r_{34}(2) &= .04696 < r_{34}(3)
\end{align*}
```

But when there is high noise, e.g., $\phi = .8$ ($p=.44$):
\begin{align*}
r_{34}(3) &= .00378\\
r_{34}(2) &= .00402 > r_{34}(3)
\end{align*}

Note that the same example works for selecting 1 winner out of the 4 candidates, as the repeated insertion model can be run in reverse.

\end{proof}

Expanded PaperInterface specification

```
∃ (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2),
  M.center = Equiv.refl (AppliedModelingLib.SocialChoice.Ranking.Candidate 2) ∧
    M.q = GGSG19TopThree.mallowsHighNoisePhi ∧
      ∃ (better : Fin 3),
        better ≠ 2 ∧
          GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.mallowsW3StaticKApprovalPairRate M 2) <
            GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.mallowsW3StaticKApprovalPairRate M better)
```

Retained governing declarations

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.finiteOutcomeLearningRate
- GGSG19TopThree.mallowsHighNoisePhi
- GGSG19TopThree.mallowsW3StaticKApprovalPairRate

Lean-elaborated claim roles

```
1. conclusion: ∃ (M : AppliedModelingLib.SocialChoice.Ranking.MallowsSpec 2),
  M.center = Equiv.refl (AppliedModelingLib.SocialChoice.Ranking.Candidate 2) ∧
    M.q = GGSG19TopThree.mallowsHighNoisePhi ∧
      ∃ (better : Fin 3),
        better ≠ 2 ∧
          GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.mallowsW3StaticKApprovalPairRate M 2) <
            GGSG19TopThree.finiteOutcomeLearningRate (GGSG19TopThree.mallowsW3StaticKApprovalPairRate M better)
```

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_theorem_lem_mallowsnotWK_counterexample

Recorded source-to-Spec screening **Verdict:** matches

Reason: The target is the source four-candidate, three-winner high-noise counterexample: it fixes the reference ranking and printed high-noise Mallows parameter, then exhibits a cutoff better than W-Approval in the same finite outcome rate.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 12

Source locator: cited publication, lines 728-734

Verbatim source input

```
\begin{restatable}{corollary}{lemmallowsnorando}
    \label{lem:mallowsnorando}
        Randomization among $K$-Approval mechanisms does not improve the learning rate for selecting $W$ winners from the Mallows model.
        ↪ For any randomized $K$-Approval mechanism $(B,D)$, where $\beta_p\in B$ corresponds to $p$-Approval, for selecting $W$ winners
        ↪ from the Mallows model, there exists an Approval rate optimal mechanism $K^*$-Approval such that $r(K^*) \geq r(B,D)$.

\end{restatable}

The proof simply notes that under the Mallows model with this goal, the candidate pair $W, W+1$ (when candidates are indexed according to reference
↪ distribution $\sigma^*$) is pivotal regardless of the $K$-Approval mechanism used.

[Next verbatim source excerpt]

\lemmallowsnorando*
\begin{proof}
    When selecting $W$ winners out of $M$ candidates, we need to separate candidates $1 \dots W$ from candidates $W + 1 \dots M$. It is easy to show
    ↪ that the pivotal pair, regardless of which $K$ is used in $K$-Approval, is $W$,$W+1$. Applying
    ↪ Theorem~\ref{lem:randomizenotbetterapproval}, then, randomization cannot help the overall rate.
\end{proof}
```

Expanded PaperInterface specification

```
∀ {n : ℕ} (center : AppliedModelingLib.SocialChoice.Ranking.Ranking n) (q : ℝ)
  (W : AppliedModelingLib.SocialChoice.Ranking.Candidate n)
  (hDomain : GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain q W) (K : ℕ),
  0 < K →
    K < n + 2 →
      GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
        (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
          (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
        W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) K =
       GGSG19TopThree.mallowsTopWKApprovalPairRate
        (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
          (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
        W K
        (GGSG19TopThree.topWBoundaryPair center W
          (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain))
```

Retained governing declarations

- AppliedModelingLib.SocialChoice.Ranking.Candidate
- AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ
- AppliedModelingLib.SocialChoice.Ranking.Ranking
- GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain
- GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
- GGSG19TopThree.mallowsTopWKApprovalPairRate
- GGSG19TopThree.topWBoundaryPair

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: AppliedModelingLib.SocialChoice.Ranking.Ranking n
3. parameter: ℝ
4. parameter: AppliedModelingLib.SocialChoice.Ranking.Candidate n
5. assumption: GGSG19TopThree.assumption_mallows_nontrivial_winner_and_parameter_domain q W
6. parameter: ℕ
7. assumption: 0 < K
8. assumption: K < n + 2
9. conclusion: GGSG19TopThree.mallowsTopWKApprovalOutcomeRate
   (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
     (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
   W (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain) K =
  GGSG19TopThree.mallowsTopWKApprovalPairRate
   (AppliedModelingLib.SocialChoice.Ranking.MallowsSpec.ofQ center q
     (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_1 q W hDomain))
   W K
   (GGSG19TopThree.topWBoundaryPair center W
     (GGSG19TopThree.PaperInterface.source_corollary_lem_mallowsnorandoSpec._proof_2 q W hDomain))

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proof_mallows_common_pivotal_pair

Recorded source-to-Spec screening **Verdict:** matches
Reason: The source proof identifies the W,W+1 pair as pivotal for every K-Approval cutoff in Mallows W-winner selection. The target states the corresponding equality between the overall outcome rate and that boundary-pair rate on the declared Mallows domain.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 13

Source locator: cited publication, lines 1544-1548

Verbatim source input

Let the items in the reference ranking be, in order, $i=1, 2, 3, 4$, respectively.

A Mallows model (with parameter $\phi = \frac{p}{1-p}$, where p is the probability of flipping a given pair of candidates) can be sampled by repeated insertion \citep{lu14a,diaconis_group_nodate}: starting from the first item in the reference ranking, there exists probability $\tilde{p}_{ij} = \frac{\phi^{i-j}}{1 + \phi^1 + \dots + \phi^{i-1}}$ at which item i can be inserted into position $j\leq i$, \textit{independently} of how items above it were inserted, such that the resulting ranking distribution matches the Mallows model.

Using this repeated insertion property for our example, we can derive $p_{\ell k}$, the probability at which item 3 is in position ℓ and item 4 is in position k after sampling from a Mallows model with parameter ϕ.

Expanded PaperInterface specification

```
∀ {N : ℕ} (q : ℝ) (hDomain : GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain q)
  (stage inserted : Fin N),
  ((GGSG19TopThree.mallowsInsertionPMF q
        (GGSG19TopThree.PaperInterface.source_proof_mallows_four_candidate_joint_location_matrixSpec._proof_1 q
          hDomain)
        stage)
      inserted).toReal =
    if ↑inserted ≤ ↑stage then q ^ (↑stage - ↑inserted) / ∑ r ∈ Finset.range (↑stage + 1), q ^ (↑stage - r) else 0
```

Retained governing declarations

- GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain
- GGSG19TopThree.mallowsInsertionPMF

Lean-elaborated claim roles

```
1. parameter: ℕ
2. parameter: ℝ
3. assumption: GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain q
4. parameter: Fin N
5. parameter: Fin N
6. conclusion: ((GGSG19TopThree.mallowsInsertionPMF q
       (GGSG19TopThree.PaperInterface.source_proof_mallows_four_candidate_joint_location_matrixSpec._proof_1 q hDomain)
       stage)
     inserted).toReal =
  if ↑inserted ≤ ↑stage then q ^ (↑stage - ↑inserted) / ∑ r ∈ Finset.range (↑stage + 1), q ^ (↑stage - r) else 0
```

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_mallows_repeated_insertion_probability_formula

Recorded source-to-Spec screening **Verdict:** matches

Reason: The atom deliberately selects only the source displayed one-stage insertion kernel. The target gives its normalized phi^(i-j) probability for each admissible position and zero outside the stated support; the adjacent independence and full ranking-law prose is preserved as source-model context rather than overstated by this formula target.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 14

Source locator: cited publication, lines 1548-1568

Verbatim source input

Using this repeated insertion property for our example, we can derive $p_{\ell k}$, the probability at which item 3 is in position ℓ and item 4 is in position k after sampling from a Mallows model with parameter ϕ.

In particular, if $\ell < k$, $p_{\ell k}$ is exactly the probability that item 3 is inserted in position ℓ and item 4 is inserted in position k. If $k>\ell$, however, it is the probability that item 3 is inserted in position $\ell - 1$ and then pushed down when item 4 is inserted in position k. (More generally, it turns out, the exact probability for an item appearing in a given position in the Mallows model can be calculated using a simple dynamic program, a fact that does not appear to be documented elsewhere but may be independently useful. We used this dynamic program to find this given example).

Then, for our example
\begin{align*}
N_i &\triangleq 1 + \phi^1 + \dots + \phi^{i-1} \\
p_{\ell k} &= \begin{bmatrix}
0 & \frac{\phi^2}{N_3}\frac{\phi^2}{N_4} & \frac{\phi^2}{N_3}\frac{\phi^1}{N_4} & \frac{\phi^2}{N_3}\frac{\phi^0}{N_4}\\
\frac{\phi^2}{N_3}\frac{\phi^3}{N_4} & 0 & \frac{\phi^1}{N_3}\frac{\phi^1}{N_4} & \frac{\phi^1}{N_3}\frac{\phi^0}{N_4}\\
\frac{\phi^1}{N_3}\frac{\phi^3}{N_4} & \frac{\phi^1}{N_3}\frac{\phi^2}{N_4} & 0 & \frac{\phi^0}{N_3}\frac{\phi^0}{N_4}\\
\frac{\phi^0}{N_3}\frac{\phi^3}{N_4} & \frac{\phi^0}{N_3}\frac{\phi^2}{N_4} & \frac{\phi^0}{N_3}\frac{\phi^1}{N_4} & 0\\
\end{bmatrix}_{\ell k}
=
\frac{1}{N_3N_4}\begin{bmatrix}
0 & \phi^4 & \phi^3 & {\phi^2}\\
\phi^5 & 0 & \phi^2 & {\phi^1}\\
\phi^4 & \phi^3 & 0 & 1\\
{\phi^3} & {\phi^2} & {\phi^1} & 0\\
\end{bmatrix}_{\ell k}
\end{align*}

Expanded PaperInterface specification

```
∀ (q : ℝ) (hDomain : GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain q) (ell k : Fin 4),
  GGSG19TopThree.ProofBridge.mallowsFourCandidate34JointLocationProb q
      (GGSG19TopThree.PaperInterface.source_proof_mallows_four_candidate_joint_location_matrixSpec._proof_1 q hDomain)
      ell k =
    if ell = k then 0
    else (q ^ if ↑ell < ↑k then 5 - ↑ell - ↑k else 6 - ↑ell - ↑k) / (1 + q + q ^ 2) / (1 + q + q ^ 2 + q ^ 3)
```

Retained governing declarations

- GGSG19TopThree.ProofBridge.mallowsFourCandidate34JointLocationProb
- GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain

Lean-elaborated claim roles

1. parameter: ℝ
2. assumption: GGSG19TopThree.assumption_mallows_repeated_insertion_parameter_domain q
3. parameter: Fin 4
4. parameter: Fin 4
5. conclusion: GGSG19TopThree.ProofBridge.mallowsFourCandidate34JointLocationProb q
   (GGSG19TopThree.PaperInterface.source_proof_mallows_four_candidate_joint_location_matrixSpec._proof_1 q hDomain) ell
   k =
  if ell = k then 0
  else (q ^ if ↑ell < ↑k then 5 - ↑ell - ↑k else 6 - ↑ell - ↑k) / (1 + q + q ^ 2) / (1 + q + q ^ 2 + q ^ 3)

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_proof_mallows_four_candidate_joint_location_matrix

Recorded source-to-Spec screening **Verdict:** matches

Reason: The source gives the complete off-diagonal joint-location matrix for reference items three and four with normalizers N_3 and N_4. The target is its single index-level matrix formula, including the zero diagonal and the same powers of phi.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 15

Source locator: cited publication, lines 1606-1656

Verbatim source input

```
\lemrandomizebetterapprovalWselection*
\begin{proof}
We provide two proofs: a numeric example from a real-world election, and a contrived, constructed example.

\parbold{Numeric example found in a real election} In Durham Ward 1, to select 2 winners, randomizing between 3 and 4-Approval is better than either
↪ individually, even though asymptotically the mechanisms pick the same set of winners. The critical pair with $2$-Approval is with the candidate
↪ asymptotically ranked 1st, and the best item not selected. With $3$-Approval, it is with the candidate asymptotically ranked 2nd, and the same best
↪ item not selected.

We will call these items $h$, $i,j$ (the one not selected) respectively. The respective probabilities of being selected alone:

%
%
%
%
%
%
%
%
%
%
%

\begin{align*}
t_{hj}^h(3)&= 0.277 \\
t_{hj}^h(4)&= 0.266       \\
t_{hj}^j(3)&= 0.200       \\
t_{hj}^j(4)&= 0.188       \\
t_{ij}^i(3)&= 0.255       \\
t_{ij}^i(4)&= 0.295       \\
t_{ij}^j(3)&= 0.160\\
t_{ij}^j(4)&= 0.217\\\\
%
t_{hj}^h(\{3, 4\})&= 0.271        \\
t_{hj}^j(\{3, 4\})&= 0.194       \\
t_{ij}^i(\{3, 4\})&= 0.275       \\
t_{ij}^j(\{3, 4\})&= 0.189\\
\end{align*}

And the resulting rates (using the formula in Proposition~\ref{lem:pairwiselearning_approval}) are:

\begin{align*}
r_{hj}(3) &= .00616932\\
r_{ij}(3) &= .01114061\\
r_{hj}(4) &= .00677352\\
r_{ij}(4) &= .00592327\\
r_{hj}(\{3, 4\}) &= .00642839\\
r_{ij}(\{3, 4\}) &= .00815633\\
r(3) &= \min(r_{hj}(3), r_{ij}(3)) = .00616932\\
r(4) &= \min(r_{hj}(4), r_{ij}(4)) = .00592327\\
r(\{3, 4\}) &= \min(r_{hj}(\{3, 4\}), r_{ij}(\{3, 4\})) = .00642839 > \max(r(3), r(4))
\end{align*}

Thus randomization improves learning.
```

Expanded PaperInterface specification

```
max (GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.durhamWard1K3Rate)
    (GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.durhamWard1K4Rate) <
  GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.durhamWard1RandomizedRate
```

Retained governing declarations

- GGSG19TopThree.DurhamWard1Pair
- GGSG19TopThree.durhamWard1K3Rate
- GGSG19TopThree.durhamWard1K4Rate
- GGSG19TopThree.durhamWard1RandomizedRate
- GGSG19TopThree.finiteOutcomeLearningRate

Lean-elaborated claim roles

```
1. conclusion: Real.lt†
  (max (GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.durhamWard1K3Rate)
    (GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.durhamWard1K4Rate))
  (GGSG19TopThree.finiteOutcomeLearningRate GGSG19TopThree.durhamWard1RandomizedRate)
```

Lean proof endpoint (built separately)

GGSG19TopThree.PaperInterface.source_example_durham_ward1_randomized_approval_improves_static3_static4

Recorded source-to-Spec screening **Verdict:** matches

Reason: The source numerical example concludes that the randomized 3/4-Approval mechanism has a larger outcome learning rate than either static choice. The target is exactly max(static K=3, static K=4) below the named randomized rate; printed probability tables remain empirical/proof context, not an implementation claim.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation