Human review packet: SeshadriUgander2020IIATesting

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

Paper: SeshadriUgander2020IIATesting
Paper id: SeshadriUgander2020IIATesting
Source version: arXiv:2001.07042v1, 20 January 2020
Source record: audit/paper_statement_map.json
Rows: 11
Generated: 2026-09-06
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 SeshadriUgander2020IIATesting --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 (28; supporting context)
- Semantic library prerequisites (11; not paper claims)
  - AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking
  - AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking.usedEdges
  - AppliedModelingLib.Foundations.Graph.degreeCount
  - AppliedModelingLib.Foundations.Graph.edgeCount
  - AppliedModelingLib.Foundations.Graph.oddDegreeFinset
  - AppliedModelingLib.Foundations.Graph.oddDegreeCount
  - AppliedModelingLib.PMFAbsoluteContinuous
  - AppliedModelingLib.finiteCrossEntropy
  - AppliedModelingLib.finiteCrossEntropyExtended
  - AppliedModelingLib.finiteCrossEntropyExtendedRange
  - AppliedModelingLib.finiteEntropy
- Paper-specific semantic prerequisites (19; not paper claims)
  - SeshadriUgander2020IIATesting.ChoiceFrame
  - SeshadriUgander2020IIATesting.ChoiceFrame.Eulerian
  - SeshadriUgander2020IIATesting.ChoiceFrame.Observation
  - SeshadriUgander2020IIATesting.ChoiceFrame.incidenceGraph
  - SeshadriUgander2020IIATesting.ChoiceSystem
  - SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign
  - SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
  - SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.AlternatingCycleWitness
  - SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean
  - SeshadriUgander2020IIATesting.ChoiceSystem.SatisfiesIIA
  - SeshadriUgander2020IIATesting.ChoiceSystem.totalVariation
  - SeshadriUgander2020IIATesting.ChoiceSystem.SeparatedFromIIA
  - SeshadriUgander2020IIATesting.FiniteDistribution.Test

  - SeshadriUgander2020IIATesting.FiniteDistribution.binaryError
  - SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest
  - SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound
  - SeshadriUgander2020IIATesting.ChoiceSystem.perturb
  - SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion
  - SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts
- Source claims (11)
  - lemma3_chiSquare_mixtureSpec
  - lemma4_cycle_chiSquareSpec
  - theorem1_testing_lower_boundSpec
  - corollary1_global_lower_bound_correctedSpec
  - fact_convex_hull_denseSpec
  - lemma_iia_projection_invarianceSpec
  - fact_entropy_cross_entropySpec
  - lemma_bipartite_cycle_decompositionSpec
  - lemma_comparison_incidence_decomposition_correctedSpec
  - corollary_all_even_subsets_correctedSpec
  - lemma2_testing_reductionSpec

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.

SeshadriUgander2020IIATesting.AllEvenChoiceSet

Declaration location: paper-local

Lean declaration body

type:
ℕ → Type

value:
fun (n : ℕ) => { s : Finset (Fin n) // 2 ≤ s.card ∧ Even s.card }

SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame

Declaration location: paper-local

Lean declaration body

type:
(n : ℕ) → 2 ≤ n → SeshadriUgander2020IIATesting.ChoiceFrame

value:
fun (n : ℕ) (hn : 2 ≤ n) =>
  { Item := Fin n, instFintypeItem := inferInstance, instDecidableEqItem := inferInstance,
    SetId := SeshadriUgander2020IIATesting.AllEvenChoiceSet n, instFintypeSetId := inferInstance,
    instDecidableEqSetId := inferInstance, members := fun (C : SeshadriUgander2020IIATesting.AllEvenChoiceSet n) => ↑C,
    nonempty_sets := SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame._proof_1 n hn,
    card_two_le := SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame._proof_2 n }

Direct retained dependencies

- SeshadriUgander2020IIATesting.AllEvenChoiceSet
- SeshadriUgander2020IIATesting.ChoiceFrame

SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount

Declaration location: paper-local

Lean declaration body

type:
SeshadriUgander2020IIATesting.ChoiceFrame → ℕ

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) => ∑ C : F.SetId, (F.members C).card

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame

SeshadriUgander2020IIATesting.ChoiceFrame.instDecidableEqObservation

Declaration location: paper-local

Lean declaration body

type:
(F : SeshadriUgander2020IIATesting.ChoiceFrame) → (a b : F.Observation) → Decidable (a = b)

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) (a b : F.Observation) => inferInstance a b

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation

SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation

Declaration location: paper-local

Lean declaration body

type:
(F : SeshadriUgander2020IIATesting.ChoiceFrame) → Fintype F.Observation

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) => Sigma.instFintype

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation

SeshadriUgander2020IIATesting.ChoiceFrame.occurrences

Declaration location: paper-local

Lean declaration body

type:
(F : SeshadriUgander2020IIATesting.ChoiceFrame) → F.Item → Finset F.SetId

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) (x : F.Item) => {C : F.SetId | x ∈ F.members C}

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame

SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → ℝ

value:
```
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} =>
  min
    (4 * Real.logb 2 ↑(Fintype.card F.Item) +
      4 * ↑(Fintype.card F.Item) * (2 * ↑(Fintype.card F.Item) - 4 * Real.logb 2 ↑(Fintype.card F.Item)) /
        ↑F.incidenceCount)
    (2 * ↑(Fintype.card F.Item))
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount

SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → ℝ

value:
```
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} =>
  min
    (↑F.incidenceCount * (4 * Real.logb 2 ↑(Fintype.card F.Item)) /
      (↑F.incidenceCount - 4 * ↑(Fintype.card F.Item) + 2 * (4 * Real.logb 2 ↑(Fintype.card F.Item))))
    (2 * ↑(Fintype.card F.Item))
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount

SeshadriUgander2020IIATesting.ChoiceSystem.finiteMixture

Declaration location: paper-local

Lean declaration body

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem F →
    (F.Observation → SeshadriUgander2020IIATesting.ChoiceSystem F) → SeshadriUgander2020IIATesting.ChoiceSystem F

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (weight : SeshadriUgander2020IIATesting.ChoiceSystem F)
    (component : F.Observation → SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  { mass := fun (o : F.Observation) => ∑ i : F.Observation, weight.mass i * (component i).mass o,
    nonneg := SeshadriUgander2020IIATesting.ChoiceSystem.finiteMixture._proof_1 weight component,
    sum_one := SeshadriUgander2020IIATesting.ChoiceSystem.finiteMixture._proof_2 weight component }
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem

SeshadriUgander2020IIATesting.ChoiceSystem.itemMass

Declaration location: paper-local

Lean declaration body

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → F.Item → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) (x : F.Item) =>
  ∑ C : { C : F.SetId // x ∈ F.members C },
    q.mass ⟨↑C, ⟨x, SeshadriUgander2020IIATesting.ChoiceSystem.itemMass._proof_1 x C⟩⟩
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem

SeshadriUgander2020IIATesting.ChoiceSystem.klDivergence

Declaration location: paper-local

Lean declaration body

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem F → SeshadriUgander2020IIATesting.ChoiceSystem F → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q p : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  ∑ o : F.Observation, q.mass o * (Real.log (q.mass o) - Real.log (p.mass o))
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem

SeshadriUgander2020IIATesting.ChoiceSystem.setMass

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → F.SetId → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) (C : F.SetId) =>
  ∑ x : ↥(F.members C), q.mass (C, x)

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem

SeshadriUgander2020IIATesting.ChoiceSystem.PositiveSetMass

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → Prop

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  ∀ (C : F.SetId), 0 < q.setMass C

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.setMass

SeshadriUgander2020IIATesting.ChoiceSystem.IIAInterior

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → Prop

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (p : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  p.SatisfiesIIA ∧ p.PositiveSetMass

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.PositiveSetMass
- SeshadriUgander2020IIATesting.ChoiceSystem.SatisfiesIIA

SeshadriUgander2020IIATesting.ChoiceSystem.InFiniteConvexHullIIA

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → Prop

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  ∃ (weight : SeshadriUgander2020IIATesting.ChoiceSystem F) (component :
    F.Observation → SeshadriUgander2020IIATesting.ChoiceSystem F),
    (∀ (i : F.Observation), (component i).SatisfiesIIA) ∧ q = weight.finiteMixture component

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.SatisfiesIIA
- SeshadriUgander2020IIATesting.ChoiceSystem.finiteMixture

SeshadriUgander2020IIATesting.ChoiceSystem.iiaProjection

Declaration location: paper-local

Lean declaration body

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem F → SeshadriUgander2020IIATesting.ChoiceSystem F → Prop

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q a : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  {p : SeshadriUgander2020IIATesting.ChoiceSystem F |
     p.IIAInterior ∧
       ∀ (r : SeshadriUgander2020IIATesting.ChoiceSystem F), r.IIAInterior → q.klDivergence p ≤ q.klDivergence r}
    a
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.IIAInterior
- SeshadriUgander2020IIATesting.ChoiceSystem.klDivergence

SeshadriUgander2020IIATesting.ChoiceSystem.signInner

Declaration location: paper-local

Lean declaration body

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F →
    SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame}
    (b b' : SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F) =>
  ∑ o : F.Observation, b.sign o * b'.sign o
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign

SeshadriUgander2020IIATesting.ChoiceSystem.uniform

Declaration location: paper-local

Lean declaration body

```
type:
(F : SeshadriUgander2020IIATesting.ChoiceFrame) → SeshadriUgander2020IIATesting.ChoiceSystem F

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) =>
  { mass := fun (x : F.Observation) => 1 / ↑F.incidenceCount,
    nonneg := SeshadriUgander2020IIATesting.ChoiceSystem.uniform._proof_1 F,
    sum_one := SeshadriUgander2020IIATesting.ChoiceSystem.uniform._proof_2 F }
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem

SeshadriUgander2020IIATesting.FiniteDistribution

Declaration location: paper-local

Lean declaration body

```
field mass:
{α : Type} → [inst : Fintype α] → SeshadriUgander2020IIATesting.FiniteDistribution α → α → ℝ

field nonneg:
∀ {α : Type} [inst : Fintype α] (self : SeshadriUgander2020IIATesting.FiniteDistribution α) (x : α), 0 ≤ self.mass x

field sum_one:
∀ {α : Type} [inst : Fintype α] (self : SeshadriUgander2020IIATesting.FiniteDistribution α), ∑ x : α, self.mass x = 1
```

SeshadriUgander2020IIATesting.ChoiceSystem.asFiniteDistribution

Declaration location: paper-local

Lean declaration body

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem F → SeshadriUgander2020IIATesting.FiniteDistribution F.Observation

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  { mass := q.mass, nonneg := q.nonneg, sum_one := q.sum_one }
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.FiniteDistribution

SeshadriUgander2020IIATesting.FiniteDistribution.chiSquare

Declaration location: paper-local

Lean declaration body

```
type:
{α : Type} →
  [inst : Fintype α] →
    SeshadriUgander2020IIATesting.FiniteDistribution α → SeshadriUgander2020IIATesting.FiniteDistribution α → ℝ

value:
fun {α : Type} [Fintype α] (p q : SeshadriUgander2020IIATesting.FiniteDistribution α) =>
  ∑ x : α, p.mass x ^ 2 / q.mass x - 1
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.FiniteDistribution

SeshadriUgander2020IIATesting.FiniteDistribution.rejectionSet

Declaration location: paper-local

Lean declaration body

type:
{α : Type} → [Fintype α] → SeshadriUgander2020IIATesting.FiniteDistribution.Test α → Finset α

value:
fun {α : Type} [Fintype α] (φ : SeshadriUgander2020IIATesting.FiniteDistribution.Test α) => {x : α | φ x = true}

Direct retained dependencies

- SeshadriUgander2020IIATesting.FiniteDistribution.Test

SeshadriUgander2020IIATesting.FiniteDistribution.sampleMass

Declaration location: paper-local

Lean declaration body

type:
{α : Type} → (α → ℝ) → (N : ℕ) → (Fin N → α) → ℝ

value:
fun {α : Type} (mass : α → ℝ) (N : ℕ) (sample : Fin N → α) => ∏ i : Fin N, mass (sample i)

SeshadriUgander2020IIATesting.FiniteDistribution.product

Declaration location: paper-local

Lean declaration body

type:
{α : Type} →
  [inst : Fintype α] →
    [DecidableEq α] →
      SeshadriUgander2020IIATesting.FiniteDistribution α →
        (N : ℕ) → SeshadriUgander2020IIATesting.FiniteDistribution (Fin N → α)

value:
fun {α : Type} [Fintype α] [DecidableEq α] (q : SeshadriUgander2020IIATesting.FiniteDistribution α) (N : ℕ) =>
  { mass := SeshadriUgander2020IIATesting.FiniteDistribution.sampleMass q.mass N,
    nonneg := SeshadriUgander2020IIATesting.FiniteDistribution.product._proof_1 q N,
    sum_one := SeshadriUgander2020IIATesting.FiniteDistribution.product._proof_2 q N }

Direct retained dependencies

- SeshadriUgander2020IIATesting.FiniteDistribution
- SeshadriUgander2020IIATesting.FiniteDistribution.sampleMass

SeshadriUgander2020IIATesting.FiniteDistribution.uniformMixture

Declaration location: paper-local

Lean declaration body

type:
{α : Type} →
  [inst : Fintype α] →
    {β : Type} →
      (B : Finset β) →
        B.Nonempty →
          (β → SeshadriUgander2020IIATesting.FiniteDistribution α) → SeshadriUgander2020IIATesting.FiniteDistribution α

value:
fun {α : Type} [Fintype α] {β : Type} (B : Finset β) (hB : B.Nonempty)
    (q : β → SeshadriUgander2020IIATesting.FiniteDistribution α) =>
  { mass := fun (x : α) => (∑ b ∈ B, (q b).mass x) / ↑B.card,
    nonneg := SeshadriUgander2020IIATesting.FiniteDistribution.uniformMixture._proof_1 B hB q,
    sum_one := SeshadriUgander2020IIATesting.FiniteDistribution.uniformMixture._proof_2 B hB q }

Direct retained dependencies

- SeshadriUgander2020IIATesting.FiniteDistribution

SeshadriUgander2020IIATesting.Rademacher.sign

Declaration location: paper-local

Lean declaration body

type:
Bool → ℝ

value:
fun (a : Bool) => if a = true then 1 else -1

SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.orientationValue

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) → (D.Cycle → Bool) → F.Observation → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame}
    (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) (a : D.Cycle → Bool) (o : F.Observation) =>
  D.base o * SeshadriUgander2020IIATesting.Rademacher.sign (a (D.edgeEquiv.symm o).fst)

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.Rademacher.sign

SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.orientedSign

Declaration location: paper-local

Lean declaration body

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) →
    (D.Cycle → Bool) → SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame}
    (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) (a : D.Cycle → Bool) =>
  { sign := D.orientationValue a,
    sign_eq_one_or_neg_one :=
      SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.orientationValue_eq_one_or_neg_one D a,
    set_balance := D.set_balance a, item_balance := D.item_balance a }

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.orientationValue

Material library prerequisites

AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking

Source locator: cited publication, lines 867-869

Verbatim paper-source connection

\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of vertices of odd degree on the second vertex set.
\end{lemma}

[Next verbatim source excerpt]

The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph G with some edges remaining.

First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from V_1 and of degree 1 or less from V_2. Second, perform a breadth-first search (BFS) starting from any remaining vertex in V_1 until a cycle σ_1 is found. Add the cycle to the collection of cycles σ, and remove the edges from G. Repeat these two steps, adding cycles σ_i to σ until no edges remain in G.

Now, we can analyze this procedure. Note that the only edges removed from G that are also not added to σ are from vertices of degree 2 or less from V_1 and from vertices of degree 1 or less from V_2. This fact allows us to upper bound the number of edges not contained in the cycle decomposition by $2n_1 + n_2$.

Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from V_1, V_1 has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from V_2, V_2 only has vertices of minimum degree 2. Thus, the arbitrary vertex in V_1 at the start of the BFS is connected to at least three vertices in V_2. Those three verticies, are in turn connected to at least one new vertex each in V_1 if a cycle is not found, all of which are in turn connected to two new vertices each in V_2 if a cycle is not found, and so on. Thus, as long as we don't find a cycle, the BFS tree grows at least as a complete binary tree does every time we go to V_2. Since there are no leaf nodes in this graph (no vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$, twice the depth of a complete binary tree since that is the maximum depth before all nodes in V_1 are visited and a cycle has to be found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2 \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma \ref{lemma:general_cycle_decomposition} is using V_2 as a bridge back to V_1 and using V_1 only to exponentially grow the BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of vertices of V_1, which is useful when the vertices are highly imbalanced in the two parts.

Since both steps can always finish as long as G has edges and the second always strictly reduces the number of edges of G, the algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most $2n_1 + n_2$ edges that remain.

We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in V_2 only if those vertices are degree 1. Thus, vertices in V_2 must either begin with odd parity in order for one of their edges to not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices from V_1 in the first step. Since vertices removed from V_1 in the first step can have degree at most 2, their removal changes the parity of at most $2n_1$ vertices in V_2. This argument yields an alternate upper bound of edges removed that do not contribute to the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.

Lean declaration type (metadata)

```
field Cycle:
{V : Type u_1} →
  [inst : DecidableEq V] →
    {G : SimpleGraph V} →
      {maxLength : ℕ} → AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength → Type

field instFintypeCycle:
{V : Type u_1} →
  [inst : DecidableEq V] →
    {G : SimpleGraph V} →
      {maxLength : ℕ} →
        (self : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength) → Fintype self.Cycle

field instDecidableEqCycle:
{V : Type u_1} →
  [inst : DecidableEq V] →
    {G : SimpleGraph V} →
      {maxLength : ℕ} →
        (self : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength) → DecidableEq self.Cycle

field walk:
{V : Type u_1} →
  [inst : DecidableEq V] →
    {G : SimpleGraph V} →
      {maxLength : ℕ} →
        (self : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength) →
          self.Cycle → (start : V) × G.Walk start start

field isCycle:
∀ {V : Type u_1} [inst : DecidableEq V] {G : SimpleGraph V} {maxLength : ℕ}
  (self : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength) (cycle : self.Cycle),
  (self.walk cycle).snd.IsCycle
```

```
field length_le:
∀ {V : Type u_1} [inst : DecidableEq V] {G : SimpleGraph V} {maxLength : ℕ}
  (self : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength) (cycle : self.Cycle),
  (self.walk cycle).snd.length ≤ maxLength

field pairwiseDisjoint:
∀ {V : Type u_1} [inst : DecidableEq V] {G : SimpleGraph V} {maxLength : ℕ}
  (self : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength),
  Set.univ.PairwiseDisjoint fun (cycle : self.Cycle) => (self.walk cycle).snd.edges.toFinset
```

Exact Lean library declaration

```
/-- A finite, edge-disjoint family of actual simple cycles in a graph. -/
structure PartialSimpleCyclePacking (G : SimpleGraph V) (maxLength : Nat) where
  Cycle : Type
  instFintypeCycle : Fintype Cycle
  instDecidableEqCycle : DecidableEq Cycle
  walk : (cycle : Cycle) → Σ start : V, G.Walk start start
  isCycle : ∀ cycle, (walk cycle).2.IsCycle
  length_le : ∀ cycle, (walk cycle).2.length ≤ maxLength
  pairwiseDisjoint : Set.PairwiseDisjoint Set.univ
    (fun cycle => (walk cycle).2.edges.toFinset)
```

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

Reason: The source lemma constructs an edge-disjoint family of simple cycles with a maximum cycle length and a residual-edge budget. This finite packing structure states exactly that cycle family, its length cap, and pairwise edge disjointness.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking.usedEdges

Source locator: cited publication, lines 867-869

Verbatim paper-source connection

```
\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
        Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$
        ↪ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most
        ↪ $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of
        ↪ vertices of odd degree on the second vertex set.
    \end{lemma}
```

[Next verbatim source excerpt]

```
        The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to
        ↪ directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by
        ↪ stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph $G$ with some edges remaining.

        First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from $V_1$ and of degree 1 or less from $V_2$.
        ↪ Second, perform a breadth-first search (BFS) starting from any remaining vertex in $V_1$ until a cycle $\sigma_1$ is found. Add the
        ↪ cycle to the collection of cycles $\sigma$, and remove the edges from $G$. Repeat these two steps, adding cycles $\sigma_i$ to
        ↪ $\sigma$ until no edges remain in $G$.

        Now, we can analyze this procedure. Note that the only edges removed from $G$ that are also not added to $\sigma$ are from vertices of
        ↪ degree 2 or less from $V_1$ and from vertices of degree 1 or less from $V_2$. This fact allows us to upper bound the number of edges
        ↪ not contained in the cycle decomposition by $2n_1 + n_2$.

        Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from
        ↪ $V_1$, $V_1$ has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from $V_2$,
        ↪ $V_2$ only has vertices of minimum degree 2. Thus, the arbitrary vertex in $V_1$ at the start of the BFS is connected to at least three
        ↪ vertices in $V_2$. Those three verticies, are in turn connected to at least one new vertex each in $V_1$ if a cycle is not found, all of
        ↪ which are in turn connected to two new vertices each in $V_2$ if a cycle is not found, and so on. Thus, as long as we don't find a cycle,
        ↪ the BFS tree grows at least as a complete binary tree does every time we go to $V_2$. Since there are no leaf nodes in this graph (no
        ↪ vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$,
        ↪ twice the depth of a complete binary tree since that is the maximum depth before all nodes in $V_1$ are visited and a cycle has to be
        ↪ found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2
        ↪ \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma
        ↪ \ref{lemma:general_cycle_decomposition} is using $V_2$ as a bridge back to $V_1$ and using $V_1$ only to exponentially grow the
        ↪ BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of
        ↪ vertices of $V_1$, which is useful when the vertices are highly imbalanced in the two parts.

        Since both steps can always finish as long as $G$ has edges and the second always strictly reduces the number of edges of $G$, the
        ↪ algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most
        ↪ $2n_1 + n_2$ edges that remain.

        We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that
        ↪ removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not
        ↪ change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in
        ↪ $V_2$ only if those vertices are degree 1. Thus, vertices in $V_2$ must either begin with odd parity in order for one of their edges to
        ↪ not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices
        ↪ from $V_1$ in the first step. Since vertices removed from $V_1$ in the first step can have degree at most 2, their removal changes the
        ↪ parity of at most $2n_1$ vertices in $V_2$. This argument yields an alternate upper bound of edges removed that do not contribute to
        ↪ the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.
```

Lean-expanded library semantic target

```
type:
{V : Type u_1} →
  [inst : DecidableEq V] →
    {G : SimpleGraph V} →
      {maxLength : ℕ} → AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength → Finset (Sym2 V)

value:
fun {V : Type u_1} [DecidableEq V] {G : SimpleGraph V} {maxLength : ℕ}
    (P : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G maxLength) =>
  Finset.univ.biUnion fun (cycle : P.Cycle) => (P.walk cycle).snd.edges.toFinset
```

Exact Lean library declaration

```
/-- The original graph edges covered by the cycle family. -/
noncomputable def usedEdges : Finset (Sym2 V) :=
  Finset.univ.biUnion (fun cycle => (P.walk cycle).2.edges.toFinset)
```

Direct retained dependencies

- AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking

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

Reason: The source distinguishes edges placed in its extracted cycles from the residual edges. The declaration is exactly the finite union of the packing's cycle-edge sets, which is the source's covered-edge set.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.Foundations.Graph.degreeCount

Source locator: cited publication, lines 867-869

Verbatim paper-source connection

```
\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
        Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$
        ↪ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most
        ↪ $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of
        ↪ vertices of odd degree on the second vertex set.
    \end{lemma}
```

[Next verbatim source excerpt]

```
        The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to
        ↪ directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by
        ↪ stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph $G$ with some edges remaining.

        First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from $V_1$ and of degree 1 or less from $V_2$.
        ↪ Second, perform a breadth-first search (BFS) starting from any remaining vertex in $V_1$ until a cycle $\sigma_1$ is found. Add the
        ↪ cycle to the collection of cycles $\sigma$, and remove the edges from $G$. Repeat these two steps, adding cycles $\sigma_i$ to
        ↪ $\sigma$ until no edges remain in $G$.

        Now, we can analyze this procedure. Note that the only edges removed from $G$ that are also not added to $\sigma$ are from vertices of
        ↪ degree 2 or less from $V_1$ and from vertices of degree 1 or less from $V_2$. This fact allows us to upper bound the number of edges
        ↪ not contained in the cycle decomposition by $2n_1 + n_2$.

        Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from
        ↪ $V_1$, $V_1$ has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from $V_2$,
        ↪ $V_2$ only has vertices of minimum degree 2. Thus, the arbitrary vertex in $V_1$ at the start of the BFS is connected to at least three
        ↪ vertices in $V_2$. Those three verticies, are in turn connected to at least one new vertex each in $V_1$ if a cycle is not found, all of
        ↪ which are in turn connected to two new vertices each in $V_2$ if a cycle is not found, and so on. Thus, as long as we don't find a cycle,
        ↪ the BFS tree grows at least as a complete binary tree does every time we go to $V_2$. Since there are no leaf nodes in this graph (no
        ↪ vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$,
        ↪ twice the depth of a complete binary tree since that is the maximum depth before all nodes in $V_1$ are visited and a cycle has to be
        ↪ found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2
        ↪ \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma
        ↪ \ref{lemma:general_cycle_decomposition} is using $V_2$ as a bridge back to $V_1$ and using $V_1$ only to exponentially grow the
        ↪ BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of
        ↪ vertices of $V_1$, which is useful when the vertices are highly imbalanced in the two parts.

        Since both steps can always finish as long as $G$ has edges and the second always strictly reduces the number of edges of $G$, the
        ↪ algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most
        ↪ $2n_1 + n_2$ edges that remain.

        We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that
        ↪ removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not
        ↪ change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in
        ↪ $V_2$ only if those vertices are degree 1. Thus, vertices in $V_2$ must either begin with odd parity in order for one of their edges to
        ↪ not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices
        ↪ from $V_1$ in the first step. Since vertices removed from $V_1$ in the first step can have degree at most 2, their removal changes the
        ↪ parity of at most $2n_1$ vertices in $V_2$. This argument yields an alternate upper bound of edges removed that do not contribute to
        ↪ the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.
```

Lean-expanded library semantic target

```
type:
{V : Type u_1} → [Fintype V] → [DecidableEq V] → SimpleGraph V → V → ℕ

value:
fun {V : Type u_1} [Fintype V] [DecidableEq V] (G : SimpleGraph V) (vertex : V) => Nat.card ↑(G.neighborSet vertex)
```

Exact Lean library declaration

```
/-- The cardinality of a vertex's neighbor set, independent of a particular
finite enumeration of those neighbors. -/
noncomputable def degreeCount {V : Type*} [Fintype V] [DecidableEq V]
    (G : SimpleGraph V) (vertex : V) : ℕ := Nat.card (G.neighborSet vertex)
```

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

Reason: The bipartite-cycle source argument uses ordinary vertex degrees while pruning and counting odd vertices. This declaration is precisely the cardinality of a graph vertex’s neighbor set.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.Foundations.Graph.edgeCount

Source locator: cited publication, lines 867-869

Verbatim paper-source connection

```
\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
        Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$
        ↪ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most
        ↪ $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of
        ↪ vertices of odd degree on the second vertex set.
    \end{lemma}
```

[Next verbatim source excerpt]

```
        The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to
        ↪ directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by
        ↪ stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph $G$ with some edges remaining.

        First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from $V_1$ and of degree 1 or less from $V_2$.
        ↪ Second, perform a breadth-first search (BFS) starting from any remaining vertex in $V_1$ until a cycle $\sigma_1$ is found. Add the
        ↪ cycle to the collection of cycles $\sigma$, and remove the edges from $G$. Repeat these two steps, adding cycles $\sigma_i$ to
        ↪ $\sigma$ until no edges remain in $G$.

        Now, we can analyze this procedure. Note that the only edges removed from $G$ that are also not added to $\sigma$ are from vertices of
        ↪ degree 2 or less from $V_1$ and from vertices of degree 1 or less from $V_2$. This fact allows us to upper bound the number of edges
        ↪ not contained in the cycle decomposition by $2n_1 + n_2$.

        Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from
        ↪ $V_1$, $V_1$ has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from $V_2$,
        ↪ $V_2$ only has vertices of minimum degree 2. Thus, the arbitrary vertex in $V_1$ at the start of the BFS is connected to at least three
        ↪ vertices in $V_2$. Those three verticies, are in turn connected to at least one new vertex each in $V_1$ if a cycle is not found, all of
        ↪ which are in turn connected to two new vertices each in $V_2$ if a cycle is not found, and so on. Thus, as long as we don't find a cycle,
        ↪ the BFS tree grows at least as a complete binary tree does every time we go to $V_2$. Since there are no leaf nodes in this graph (no
        ↪ vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$,
        ↪ twice the depth of a complete binary tree since that is the maximum depth before all nodes in $V_1$ are visited and a cycle has to be
        ↪ found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2
        ↪ \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma
        ↪ \ref{lemma:general_cycle_decomposition} is using $V_2$ as a bridge back to $V_1$ and using $V_1$ only to exponentially grow the
        ↪ BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of
        ↪ vertices of $V_1$, which is useful when the vertices are highly imbalanced in the two parts.

        Since both steps can always finish as long as $G$ has edges and the second always strictly reduces the number of edges of $G$, the
        ↪ algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most
        ↪ $2n_1 + n_2$ edges that remain.

        We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that
        ↪ removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not
        ↪ change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in
        ↪ $V_2$ only if those vertices are degree 1. Thus, vertices in $V_2$ must either begin with odd parity in order for one of their edges to
        ↪ not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices
        ↪ from $V_1$ in the first step. Since vertices removed from $V_1$ in the first step can have degree at most 2, their removal changes the
        ↪ parity of at most $2n_1$ vertices in $V_2$. This argument yields an alternate upper bound of edges removed that do not contribute to
        ↪ the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.
```

Lean-expanded library semantic target

```
type:
{V : Type u_1} → [Fintype V] → [DecidableEq V] → SimpleGraph V → ℕ

value:
fun {V : Type u_1} [Fintype V] [DecidableEq V] (G : SimpleGraph V) => Nat.card ↑G.edgeSet
```

Exact Lean library declaration

```
/-- The cardinality of a finite graph's edge set, independent of a particular
decidable presentation of adjacency. -/
noncomputable def edgeCount {V : Type*} [Fintype V] [DecidableEq V]
    (G : SimpleGraph V) : ℕ := Nat.card G.edgeSet
```

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

Reason: The source writes d=|E| for the bipartite graph’s edge count. The declaration is the cardinality of that finite graph edge set.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.Foundations.Graph.oddDegreeFinset

Source locator: cited publication, lines 867-869

Verbatim paper-source connection

```
\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
        Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$
        ↪ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most
        ↪ $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of
        ↪ vertices of odd degree on the second vertex set.
    \end{lemma}
```

[Next verbatim source excerpt]

```
        The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to
        ↪ directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by
        ↪ stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph $G$ with some edges remaining.

        First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from $V_1$ and of degree 1 or less from $V_2$.
        ↪ Second, perform a breadth-first search (BFS) starting from any remaining vertex in $V_1$ until a cycle $\sigma_1$ is found. Add the
        ↪ cycle to the collection of cycles $\sigma$, and remove the edges from $G$. Repeat these two steps, adding cycles $\sigma_i$ to
        ↪ $\sigma$ until no edges remain in $G$.

        Now, we can analyze this procedure. Note that the only edges removed from $G$ that are also not added to $\sigma$ are from vertices of
        ↪ degree 2 or less from $V_1$ and from vertices of degree 1 or less from $V_2$. This fact allows us to upper bound the number of edges
        ↪ not contained in the cycle decomposition by $2n_1 + n_2$.

        Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from
        ↪ $V_1$, $V_1$ has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from $V_2$,
        ↪ $V_2$ only has vertices of minimum degree 2. Thus, the arbitrary vertex in $V_1$ at the start of the BFS is connected to at least three
        ↪ vertices in $V_2$. Those three verticies, are in turn connected to at least one new vertex each in $V_1$ if a cycle is not found, all of
        ↪ which are in turn connected to two new vertices each in $V_2$ if a cycle is not found, and so on. Thus, as long as we don't find a cycle,
        ↪ the BFS tree grows at least as a complete binary tree does every time we go to $V_2$. Since there are no leaf nodes in this graph (no
        ↪ vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$,
        ↪ twice the depth of a complete binary tree since that is the maximum depth before all nodes in $V_1$ are visited and a cycle has to be
        ↪ found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2
        ↪ \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma
        ↪ \ref{lemma:general_cycle_decomposition} is using $V_2$ as a bridge back to $V_1$ and using $V_1$ only to exponentially grow the
        ↪ BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of
        ↪ vertices of $V_1$, which is useful when the vertices are highly imbalanced in the two parts.

        Since both steps can always finish as long as $G$ has edges and the second always strictly reduces the number of edges of $G$, the
        ↪ algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most
        ↪ $2n_1 + n_2$ edges that remain.

        We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that
        ↪ removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not
        ↪ change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in
        ↪ $V_2$ only if those vertices are degree 1. Thus, vertices in $V_2$ must either begin with odd parity in order for one of their edges to
        ↪ not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices
        ↪ from $V_1$ in the first step. Since vertices removed from $V_1$ in the first step can have degree at most 2, their removal changes the
        ↪ parity of at most $2n_1$ vertices in $V_2$. This argument yields an alternate upper bound of edges removed that do not contribute to
        ↪ the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.
```

Lean-expanded library semantic target

```
type:
{V : Type u_1} → [Fintype V] → [DecidableEq V] → SimpleGraph V → Set V → Finset V

value:
fun {V : Type u_1} [Fintype V] [DecidableEq V] (G : SimpleGraph V) (right : Set V) =>
  {vertex ∈ (AppliedModelingLib.Foundations.Graph.oddDegreeFinset._proof_1 right).toFinset |
    Odd (AppliedModelingLib.Foundations.Graph.degreeCount G vertex)}
```

Exact Lean library declaration

```
/-- The finite set of vertices in `right` whose graph degree is odd. -/
noncomputable def oddDegreeFinset (G : SimpleGraph V) (right : Set V) : Finset V :=
  right.toFinite.toFinset.filter fun vertex => Odd (degreeCount G vertex)
```

Direct retained dependencies

- AppliedModelingLib.Foundations.Graph.degreeCount

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

Reason: This finite set selects exactly the right-side vertices whose graph degree is odd, the set counted as n_2^(odd) in the source bipartite decomposition lemma.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.Foundations.Graph.oddDegreeCount

Source locator: cited publication, lines 867-869

Verbatim paper-source connection

```
\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
        Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$
        ↪ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most
        ↪ $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of
        ↪ vertices of odd degree on the second vertex set.
    \end{lemma}
```

[Next verbatim source excerpt]

```
        The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to
        ↪ directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by
        ↪ stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph $G$ with some edges remaining.

        First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from $V_1$ and of degree 1 or less from $V_2$.
        ↪ Second, perform a breadth-first search (BFS) starting from any remaining vertex in $V_1$ until a cycle $\sigma_1$ is found. Add the
        ↪ cycle to the collection of cycles $\sigma$, and remove the edges from $G$. Repeat these two steps, adding cycles $\sigma_i$ to
        ↪ $\sigma$ until no edges remain in $G$.

        Now, we can analyze this procedure. Note that the only edges removed from $G$ that are also not added to $\sigma$ are from vertices of
        ↪ degree 2 or less from $V_1$ and from vertices of degree 1 or less from $V_2$. This fact allows us to upper bound the number of edges
        ↪ not contained in the cycle decomposition by $2n_1 + n_2$.

        Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from
        ↪ $V_1$, $V_1$ has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from $V_2$,
        ↪ $V_2$ only has vertices of minimum degree 2. Thus, the arbitrary vertex in $V_1$ at the start of the BFS is connected to at least three
        ↪ vertices in $V_2$. Those three verticies, are in turn connected to at least one new vertex each in $V_1$ if a cycle is not found, all of
        ↪ which are in turn connected to two new vertices each in $V_2$ if a cycle is not found, and so on. Thus, as long as we don't find a cycle,
        ↪ the BFS tree grows at least as a complete binary tree does every time we go to $V_2$. Since there are no leaf nodes in this graph (no
        ↪ vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$,
        ↪ twice the depth of a complete binary tree since that is the maximum depth before all nodes in $V_1$ are visited and a cycle has to be
        ↪ found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2
        ↪ \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma
        ↪ \ref{lemma:general_cycle_decomposition} is using $V_2$ as a bridge back to $V_1$ and using $V_1$ only to exponentially grow the
        ↪ BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of
        ↪ vertices of $V_1$, which is useful when the vertices are highly imbalanced in the two parts.

        Since both steps can always finish as long as $G$ has edges and the second always strictly reduces the number of edges of $G$, the
        ↪ algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most
        ↪ $2n_1 + n_2$ edges that remain.

        We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that
        ↪ removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not
        ↪ change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in
        ↪ $V_2$ only if those vertices are degree 1. Thus, vertices in $V_2$ must either begin with odd parity in order for one of their edges to
        ↪ not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices
        ↪ from $V_1$ in the first step. Since vertices removed from $V_1$ in the first step can have degree at most 2, their removal changes the
        ↪ parity of at most $2n_1$ vertices in $V_2$. This argument yields an alternate upper bound of edges removed that do not contribute to
        ↪ the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.
```

Lean-expanded library semantic target

```
type:
{V : Type u_1} → [Fintype V] → [DecidableEq V] → SimpleGraph V → Set V → ℕ

value:
fun {V : Type u_1} [Fintype V] [DecidableEq V] (G : SimpleGraph V) (right : Set V) =>
  (AppliedModelingLib.Foundations.Graph.oddDegreeFinset G right).card
```

Exact Lean library declaration

```
/-- The number of vertices in `right` whose graph degree is odd. -/
noncomputable def oddDegreeCount (G : SimpleGraph V) (right : Set V) : Nat :=
  (oddDegreeFinset G right).card
```

Direct retained dependencies

- AppliedModelingLib.Foundations.Graph.oddDegreeFinset

Recorded source-to-library screening

Verdict: matches
Reason: The source residual bound uses n_2^(odd), the number of right-side vertices with odd degree. The declaration is exactly that count for the selected right vertex set.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

AppliedModelingLib.PMFAbsoluteContinuous

Source locator: cited publication, lines 840-848

Verbatim paper-source connection

```
\begin{fact}\label{max_entropy}
        For any $q \in \Delta_s$,
        \begin{align}
        \inf_{p \in \Delta_s}
        -\sum_{i=1}^s q_i \log(p_i)
        =  H(q),
        \end{align}
        where $H(q)$ denotes the entropy of $q$.
    \end{fact}
```

[Next verbatim source excerpt]

```
    \begin{proof}
        \begin{align*}
        \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) &= \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) + \sum_{i=1}^s q_i \log(q_i) -
        ↪    \sum_{i=1}^s q_i \log(q_i)\\
        &= \inf_{p \in \Delta_s} \sum_{i=1}^s q_i \log\Big(\frac{q_i}{p_i}\Big)+ \sum_{i=1}^s q_i \log\Big(\frac{1}{q_i}\Big)\\
        &= \inf_{p \in \Delta_s} D_{KL}(q||p) + H(q)\\
        &=H(q),
        \end{align*}
        where the last line follows from the non-negativity of KL divergence, and it being zero if and only if $p = q$.
```

Lean-expanded library semantic target

type:
```
{α : Type u_1} → PMF α → PMF α → Prop
```

value:
```
fun {α : Type u_1} (first second : PMF α) => ∀ (outcome : α), 0 < (first outcome).toReal → 0 < (second outcome).toReal
```

Exact Lean library declaration

```
/-- Every positive-mass atom of the first PMF also has positive mass under the second PMF. -/
def PMFAbsoluteContinuous {α : Type*} (first second : PMF α) : Prop :=
  ∀ outcome, 0 < (first outcome).toReal → 0 < (second outcome).toReal
```

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

Reason: The entropy fact's KL proof distinguishes the boundary case where a positive-mass q atom is assigned zero mass by p. This predicate expresses exactly the required finite-support condition for a finite cross-entropy value.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.finiteCrossEntropy

Source locator: cited publication, lines 840-848

Verbatim paper-source connection

```
\begin{fact}\label{max_entropy}
        For any $q \in \Delta_s$,
        \begin{align}
        \inf_{p \in \Delta_s}
        -\sum_{i=1}^s q_i \log(p_i)
        =  H(q),
        \end{align}
        where $H(q)$ denotes the entropy of $q$.
    \end{fact}
```

[Next verbatim source excerpt]

```
    \begin{proof}
        \begin{align*}
        \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) &= \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) + \sum_{i=1}^s q_i \log(q_i) -
        ↪    \sum_{i=1}^s q_i \log(q_i)\\
        &= \inf_{p \in \Delta_s} \sum_{i=1}^s q_i \log\Big(\frac{q_i}{p_i}\Big)+ \sum_{i=1}^s q_i \log\Big(\frac{1}{q_i}\Big)\\
        &= \inf_{p \in \Delta_s} D_{KL}(q||p) + H(q)\\
        &=H(q),
        \end{align*}
        where the last line follows from the non-negativity of KL divergence, and it being zero if and only if $p = q$.
```

Lean-expanded library semantic target

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

value:
fun {α : Type u_1} [Fintype α] [DecidableEq α] (law reference : PMF α) =>
  -∑ outcome : α, (law outcome).toReal * Real.log (reference outcome).toReal
```

Exact Lean library declaration

```
/-- Cross entropy of a finite law against a reference law.  The definition is
real-valued for every PMF; results interpreting it through KL make the
reference's full support explicit. -/
noncomputable def finiteCrossEntropy {α : Type*} [Fintype α] [DecidableEq α]
    (law reference : PMF α) : ℝ :=
  -∑ outcome : α, (law outcome).toReal * Real.log (reference outcome).toReal
```

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

Reason: The declaration is the source objective -sum_i q_i log p_i on a finite simplex. It uses the same probability-mass coordinates and real logarithm.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.finiteCrossEntropyExtended

Source locator: cited publication, lines 840-848

Verbatim paper-source connection

```
\begin{fact}\label{max_entropy}
        For any $q \in \Delta_s$,
        \begin{align}
        \inf_{p \in \Delta_s}
        -\sum_{i=1}^s q_i \log(p_i)
        =  H(q),
        \end{align}
        where $H(q)$ denotes the entropy of $q$.
    \end{fact}
```

[Next verbatim source excerpt]

```
    \begin{proof}
        \begin{align*}
        \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) &= \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) + \sum_{i=1}^s q_i \log(q_i) -
        ↪    \sum_{i=1}^s q_i \log(q_i)\\
        &= \inf_{p \in \Delta_s} \sum_{i=1}^s q_i \log\Big(\frac{q_i}{p_i}\Big)+ \sum_{i=1}^s q_i \log\Big(\frac{1}{q_i}\Big)\\
        &= \inf_{p \in \Delta_s} D_{KL}(q||p) + H(q)\\
        &=H(q),
        \end{align*}
        where the last line follows from the non-negativity of KL divergence, and it being zero if and only if $p = q$.
```

Lean-expanded library semantic target

```
type:
{α : Type u_1} → [Fintype α] → [DecidableEq α] → PMF α → PMF α → WithTop ℝ

value:
fun {α : Type u_1} [Fintype α] [DecidableEq α] (law reference : PMF α) =>
  if AppliedModelingLib.PMFAbsoluteContinuous law reference then ↑(AppliedModelingLib.finiteCrossEntropy law reference)
  else ⊤
```

Exact Lean library declaration

```
/-- Extended-real cross entropy with the usual information-theoretic boundary
convention: a reference assigning zero mass to a positive-mass data atom has
cost `⊤`.  On absolutely continuous pairs this agrees with the finite real
sum `finiteCrossEntropy`. -/
noncomputable def finiteCrossEntropyExtended {α : Type*} [Fintype α] [DecidableEq α]
    (law reference : PMF α) : WithTop ℝ := by
  classical
  exact if PMFAbsoluteContinuous law reference then finiteCrossEntropy law reference else ⊤
```

Direct retained dependencies

- AppliedModelingLib.PMFAbsoluteContinuous
- AppliedModelingLib.finiteCrossEntropy

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

Reason: The source entropy identity has the usual cross-entropy boundary convention. This declaration makes that convention explicit by assigning top when a positive-mass law atom lacks reference support and agrees with the finite sum otherwise.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.finiteCrossEntropyExtendedRange

Source locator: cited publication, lines 840-848

Verbatim paper-source connection

```
\begin{fact}\label{max_entropy}
        For any $q \in \Delta_s$,
        \begin{align}
        \inf_{p \in \Delta_s}
        -\sum_{i=1}^s q_i \log(p_i)
        =  H(q),
        \end{align}
        where $H(q)$ denotes the entropy of $q$.
    \end{fact}
```

[Next verbatim source excerpt]

```
    \begin{proof}
        \begin{align*}
        \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) &= \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) + \sum_{i=1}^s q_i \log(q_i) -
        ↪    \sum_{i=1}^s q_i \log(q_i)\\
        &= \inf_{p \in \Delta_s} \sum_{i=1}^s q_i \log\Big(\frac{q_i}{p_i}\Big)+ \sum_{i=1}^s q_i \log\Big(\frac{1}{q_i}\Big)\\
        &= \inf_{p \in \Delta_s} D_{KL}(q||p) + H(q)\\
        &=H(q),
        \end{align*}
        where the last line follows from the non-negativity of KL divergence, and it being zero if and only if $p = q$.
```

Lean-expanded library semantic target

type:
```
{α : Type u_1} → [Fintype α] → [DecidableEq α] → PMF α → WithTop ℝ → Prop
```

value:
```
fun {α : Type u_1} [Fintype α] [DecidableEq α] (law : PMF α) (a : WithTop ℝ) =>
  {value : WithTop ℝ | ∃ (reference : PMF α), value = AppliedModelingLib.finiteCrossEntropyExtended law reference} a
```

Exact Lean library declaration

```
/-- The range of extended-real cross entropy as the reference ranges over all
finite probability mass functions. -/
noncomputable def finiteCrossEntropyExtendedRange
    {α : Type*} [Fintype α] [DecidableEq α] (law : PMF α) : Set (WithTop ℝ) :=
  { value | ∃ reference : PMF α, value = finiteCrossEntropyExtended law reference }
```

Direct retained dependencies

- AppliedModelingLib.finiteCrossEntropyExtended

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

Reason: The source takes the infimum of cross entropy as p ranges over the finite simplex. The declaration is exactly the set of those extended-real objective values whose infimum is used in the formal fact.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

AppliedModelingLib.finiteEntropy

Source locator: cited publication, lines 840-848

Verbatim paper-source connection

```
\begin{fact}\label{max_entropy}
        For any $q \in \Delta_s$,
        \begin{align}
        \inf_{p \in \Delta_s}
        -\sum_{i=1}^s q_i \log(p_i)
        =  H(q),
        \end{align}
        where $H(q)$ denotes the entropy of $q$.
    \end{fact}
```

[Next verbatim source excerpt]

```
    \begin{proof}
        \begin{align*}
        \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) &= \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) + \sum_{i=1}^s q_i \log(q_i) -
        ↪    \sum_{i=1}^s q_i \log(q_i)\\
        &= \inf_{p \in \Delta_s} \sum_{i=1}^s q_i \log\Big(\frac{q_i}{p_i}\Big)+ \sum_{i=1}^s q_i \log\Big(\frac{1}{q_i}\Big)\\
        &= \inf_{p \in \Delta_s} D_{KL}(q||p) + H(q)\\
        &=H(q),
        \end{align*}
        where the last line follows from the non-negativity of KL divergence, and it being zero if and only if $p = q$.
```

Lean-expanded library semantic target

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

value:
fun {α : Type u_1} [Fintype α] [DecidableEq α] (law : PMF α) =>
  -∑ outcome : α, (law outcome).toReal * Real.log (law outcome).toReal
```

Exact Lean library declaration

```
/-- The finite Shannon entropy, with the standard zero-mass contribution
given by Mathlib's `Real.log 0 = 0` convention. -/
noncomputable def finiteEntropy {α : Type*} [Fintype α] [DecidableEq α]
    (law : PMF α) : ℝ :=
  -∑ outcome : α, (law outcome).toReal * Real.log (law outcome).toReal
```

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

Reason: The declaration is the source Shannon entropy H(q)=-sum_i q_i log q_i on the finite simplex, with the standard zero-mass convention made explicit by the underlying real logarithm.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Paper-specific semantic prerequisites

SeshadriUgander2020IIATesting.ChoiceFrame

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceFrame
Source locator: cited publication, lines 235-246

Verbatim paper-source connection

```
\subsection{Choice systems}\label{subsection:choice_system}
    Let $\mathcal{X}$ be a finite set of $n$ \textit{items} with generic element $x \in \mathcal{X}$, and let $\mathcal{C} \subseteq 2^\mathcal{X}$
    ↪ denote the observed space, a collection of $m \geq n$ unique unordered subsets $C \subseteq \mathcal{X}$ of size 2 or greater. A decision
    ↪ maker who is presented with a \textit{choice set} $C \in \mathcal{C}$ chooses one item from the set. We let $P_{x,C}$ denote the probability
    ↪ that $x$ is chosen from $C$, $\forall x \in C$, and let $w(C)$ denote the probability of seeing choice set $C$, $\forall C \in \mathcal{C}$. The
    ↪ object $\mathcal{C}$ should be treated as a sample frame---the regime of sets of interest, or the regime of sets for which a notion of error is
    ↪ important---determined ahead of time (as opposed to being defined in terms of the sets so far observed in some dataset). That is, we are
    ↪ operating in a fixed-design setting, where $\mathcal{C}$ does not depend on, e.g., prior observations.

    Our test is focused on a given dataset $\mathcal{D}_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ that results from a decision maker making choices: a
    ↪ datapoint $j$ represents a decision scenario, and contains $C_j$, the choice set provided in that decision, and $x_j \in C_j$, the item chosen from
    ↪ that choice set. We assume each datapoint is an independent sample from the joint distribution over choices sets and choices:
    \begin{align*}
    Pr[(x_j, C_j) = (x, C)] = w(C) \cdot P_{x,C}, \ \ \forall j.
    \end{align*}
    That is, a datapoint is a sample from some discrete distribution over $d$ possible (choice, choice set) pairs, where $d = \sum_{C \in \mathcal{C}}
    ↪ |C|$. We refer to this $d$-dimensional discrete distribution as a \textit{choice system} and denote it using $q$.
    Let $\Pall$ denote the collection of all choice systems on the collection $\mathcal{C}$, i.e., the collection of all $d$-dimensional discrete
    ↪ distributions. The (choice, choice set) tuples $(x,C)$ can be used to intuitively index this distribution, so we will generally use $(x,C)$ in place of
    ↪ $1,\ldots,d$ when addressing a specific position in a specific choice system $q$.


    Our definition of a choice system differs from Falmagne's \cite{falmagne1978representation,barbera1986falmagne} related definition of a {\it
    ↪ system of choice probabilities} $\{P_{x,C}\}_{\forall C \subseteq \mathcal X, \forall x \in C}$. Our definition of a system explicitly includes the
    ↪ probability distribution over sets, $w(C)$, $\forall C \in \mathcal X$, which Falmagne's concept does not. Falmagne's definition can be thought of
    ↪ as studying choices separated from an exogenously defined collection of choice sets. Further, because our choice systems are built to span only
    ↪ observed $C \in \mathcal{C}$, under our definition all choice systems have a positive set probability $w(C)>0$ for all $C \in \mathcal{C}$.
    ↪ Without the assumption that $w(C)>0$ for all $C \in \mathcal{C}$, the worst case error of any test of IIA would be trivially large, since it has no
    ↪ chance of measuring a violation of IIA within a set $C$ that it can not observe.
```

Lean-elaborated paper signature

```
field Item:
SeshadriUgander2020IIATesting.ChoiceFrame → Type

field instFintypeItem:
(self : SeshadriUgander2020IIATesting.ChoiceFrame) → Fintype self.Item

field instDecidableEqItem:
(self : SeshadriUgander2020IIATesting.ChoiceFrame) → DecidableEq self.Item

field SetId:
SeshadriUgander2020IIATesting.ChoiceFrame → Type

field instFintypeSetId:
(self : SeshadriUgander2020IIATesting.ChoiceFrame) → Fintype self.SetId

field instDecidableEqSetId:
(self : SeshadriUgander2020IIATesting.ChoiceFrame) → DecidableEq self.SetId

field members:
(self : SeshadriUgander2020IIATesting.ChoiceFrame) → self.SetId → Finset self.Item

field nonempty_sets:
∀ (self : SeshadriUgander2020IIATesting.ChoiceFrame), Nonempty self.SetId

field card_two_le:
∀ (self : SeshadriUgander2020IIATesting.ChoiceFrame) (C : self.SetId), 2 ≤ (self.members C).card
```

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

Reason: The declaration is the source finite choice-set family: a finite item universe, finite observed sets, membership incidences, nonempty support, and the source's at-least-two-items condition. The Lean structure adds only explicit finiteness and decidable-equality data needed to represent that model.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceFrame.Eulerian

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceFrame.Eulerian
Source locator: cited publication, lines 281-291

Verbatim paper-source connection

```
\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
    To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with $n$ nodes for
    ↪ each item in $\mathcal{X}$ and $m$ nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes
    ↪ to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in $C$. This
    ↪ bipartite graph can also be thought of as representing a hypergraph with $n$ nodes, one for each item in $\mathcal{X}$, and $m$ hyperedges,
    ↪ one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain $d$ edges. As a
    ↪ result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the
    ↪ study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

    Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item $i$ to
    ↪ every item $j$.

    This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by
    ↪ Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq
    ↪ n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and
    ↪ sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we
    ↪ prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we
    ↪ shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice
    ↪ systems are IIA, rendering a test for IIA meaningless.

    We call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by
    ↪ $\sigma$. We define two functions of $\sigma$ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle
    ↪ length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion
    ↪ index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to
    ↪ a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms
    ↪ further in Section \ref{sec:bounding}.

    For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$
    ↪ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in
    ↪ $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a
    ↪ consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
    We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do
    ↪ not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes
    ↪ the requirement of even set sizes and even item appearances.
```

Lean-elaborated paper signature

```
field set_even:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame}, F.Eulerian → ∀ (C : F.SetId), Even (F.members C).card

field item_even:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame}, F.Eulerian → ∀ (x : F.Item), Even (F.occurrences x).card
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.occurrences

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

Reason: The source restriction requires every observed choice set to have even size and every item to occur an even number of times. The two Lean fields state precisely those two even-degree conditions on the comparison-incidence graph.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceFrame.Observation

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceFrame.Observation
Source locator: cited publication, lines 235-246

Verbatim paper-source connection

\subsection{Choice systems}\label{subsection:choice_system}
Let $\mathcal{X}$ be a finite set of n \textit{items} with generic element $x \in \mathcal{X}$, and let $\mathcal{C} \subseteq 2^\mathcal{X}$ denote the observed space, a collection of $m \geq n$ unique unordered subsets $C \subseteq \mathcal{X}$ of size 2 or greater. A decision maker who is presented with a \textit{choice set} $C \in \mathcal{C}$ chooses one item from the set. We let $P_{x,C}$ denote the probability that x is chosen from C, $\forall x \in C$, and let $w(C)$ denote the probability of seeing choice set C, $\forall C \in \mathcal{C}$. The object $\mathcal{C}$ should be treated as a sample frame---the regime of sets of interest, or the regime of sets for which a notion of error is important---determined ahead of time (as opposed to being defined in terms of the sets so far observed in some dataset). That is, we are operating in a fixed-design setting, where $\mathcal{C}$ does not depend on, e.g., prior observations.

Our test is focused on a given dataset $\mathcal{D}_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ that results from a decision maker making choices: a datapoint j represents a decision scenario, and contains C_j, the choice set provided in that decision, and $x_j \in C_j$, the item chosen from that choice set. We assume each datapoint is an independent sample from the joint distribution over choices sets and choices:
\begin{align*}
Pr[(x_j, C_j) = (x, C)] = w(C) \cdot P_{x,C}, \ \ \forall j.
\end{align*}
That is, a datapoint is a sample from some discrete distribution over d possible (choice, choice set) pairs, where $d = \sum_{C \in \mathcal{C}} |C|$. We refer to this d-dimensional discrete distribution as a \textit{choice system} and denote it using q.
Let $\Pall$ denote the collection of all choice systems on the collection $\mathcal{C}$, i.e., the collection of all d-dimensional discrete distributions. The (choice, choice set) tuples (x,C) can be used to intuitively index this distribution, so we will generally use (x,C) in place of $1,\ldots,d$ when addressing a specific position in a specific choice system q.

Our definition of a choice system differs from Falmagne's \cite{falmagne1978representation,barbera1986falmagne} related definition of a {\it system of choice probabilities} $\{P_{x,C}\}_{\forall C \subseteq \mathcal X, \forall x \in C}$. Our definition of a system explicitly includes the probability distribution over sets, $w(C)$, $\forall C \in \mathcal X$, which Falmagne's concept does not. Falmagne's definition can be thought of as studying choices separated from an exogenously defined collection of choice sets. Further, because our choice systems are built to span only observed $C \in \mathcal{C}$, under our definition all choice systems have a positive set probability $w(C)>0$ for all $C \in \mathcal{C}$. Without the assumption that $w(C)>0$ for all $C \in \mathcal{C}$, the worst case error of any test of IIA would be trivially large, since it has no chance of measuring a violation of IIA within a set C that it can not observe.

Lean-expanded paper semantic target

type:
SeshadriUgander2020IIATesting.ChoiceFrame → Type

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) => (C : F.SetId) × ↥(F.members C)

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame

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

Reason: The source sample space consists of item/choice-set incidences (x,C) with x in C. The dependent-pair definition is exactly that finite support and introduces no additional modeling restriction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceFrame.incidenceGraph

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceFrame.incidenceGraph
Source locator: cited publication, lines 281-291

Verbatim paper-source connection

```
\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
    To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with $n$ nodes for
    ↪ each item in $\mathcal{X}$ and $m$ nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes
    ↪ to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in $C$. This
    ↪ bipartite graph can also be thought of as representing a hypergraph with $n$ nodes, one for each item in $\mathcal{X}$, and $m$ hyperedges,
    ↪ one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain $d$ edges. As a
    ↪ result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the
    ↪ study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

    Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item $i$ to
    ↪ every item $j$.

    This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by
    ↪ Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq
    ↪ n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and
    ↪ sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we
    ↪ prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we
    ↪ shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice
    ↪ systems are IIA, rendering a test for IIA meaningless.

    We call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by
    ↪ $\sigma$. We define two functions of $\sigma$ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle
    ↪ length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion
    ↪ index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to
    ↪ a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms
    ↪ further in Section \ref{sec:bounding}.

    For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$
    ↪ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in
    ↪ $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a
    ↪ consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
    We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do
    ↪ not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes
    ↪ the requirement of even set sizes and even item appearances.
```

Lean-expanded paper semantic target

```
type:
(F : SeshadriUgander2020IIATesting.ChoiceFrame) → SimpleGraph (F.Item ⊕ F.SetId)

value:
fun (F : SeshadriUgander2020IIATesting.ChoiceFrame) =>
  {
    Adj := fun (u v : F.Item ⊕ F.SetId) =>
      match u, v with
      | Sum.inl x, Sum.inr C => x ∈ F.members C
      | Sum.inr C, Sum.inl x => x ∈ F.members C
      | x, x_1 => False,
    symm := SeshadriUgander2020IIATesting.ChoiceFrame.incidenceGraph._proof_1 F,
    loopless := SeshadriUgander2020IIATesting.ChoiceFrame.incidenceGraph._proof_2 F }
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame

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

Reason: The target is the source bipartite comparison-incidence graph: its vertices are items and choice sets, and its edges are exactly membership relations. The symmetric simple-graph representation is a direct formal encoding.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem
Source locator: cited publication, lines 235-246

Verbatim paper-source connection

```
\subsection{Choice systems}\label{subsection:choice_system}
    Let $\mathcal{X}$ be a finite set of $n$ \textit{items} with generic element $x \in \mathcal{X}$, and let $\mathcal{C} \subseteq 2^\mathcal{X}$
    ↪ denote the observed space, a collection of $m \geq n$ unique unordered subsets $C \subseteq \mathcal{X}$ of size 2 or greater. A decision
    ↪ maker who is presented with a \textit{choice set} $C \in \mathcal{C}$ chooses one item from the set. We let $P_{x,C}$ denote the probability
    ↪ that $x$ is chosen from $C$, $\forall x \in C$, and let $w(C)$ denote the probability of seeing choice set $C$, $\forall C \in \mathcal{C}$. The
    ↪ object $\mathcal{C}$ should be treated as a sample frame---the regime of sets of interest, or the regime of sets for which a notion of error is
    ↪ important---determined ahead of time (as opposed to being defined in terms of the sets so far observed in some dataset). That is, we are
    ↪ operating in a fixed-design setting, where $\mathcal{C}$ does not depend on, e.g., prior observations.

    Our test is focused on a given dataset $\mathcal{D}_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ that results from a decision maker making choices: a
    ↪ datapoint $j$ represents a decision scenario, and contains $C_j$, the choice set provided in that decision, and $x_j \in C_j$, the item chosen from
    ↪ that choice set. We assume each datapoint is an independent sample from the joint distribution over choices sets and choices:
    \begin{align*}
    Pr[(x_j, C_j) = (x, C)] = w(C) \cdot P_{x,C}, \ \ \forall j.
    \end{align*}
    That is, a datapoint is a sample from some discrete distribution over $d$ possible (choice, choice set) pairs, where $d = \sum_{C \in \mathcal{C}}
    ↪ |C|$. We refer to this $d$-dimensional discrete distribution as a \textit{choice system} and denote it using $q$.
    Let $\Pall$ denote the collection of all choice systems on the collection $\mathcal{C}$, i.e., the collection of all $d$-dimensional discrete
    ↪ distributions. The (choice, choice set) tuples $(x,C)$ can be used to intuitively index this distribution, so we will generally use $(x,C)$ in place of
    ↪ $1,\ldots,d$ when addressing a specific position in a specific choice system $q$.


    Our definition of a choice system differs from Falmagne's \cite{falmagne1978representation,barbera1986falmagne} related definition of a {\it
    ↪ system of choice probabilities} $\{P_{x,C}\}_{\forall C \subseteq \mathcal X, \forall x \in C}$. Our definition of a system explicitly includes the
    ↪ probability distribution over sets, $w(C)$, $\forall C \in \mathcal X$, which Falmagne's concept does not. Falmagne's definition can be thought of
    ↪ as studying choices separated from an exogenously defined collection of choice sets. Further, because our choice systems are built to span only
    ↪ observed $C \in \mathcal{C}$, under our definition all choice systems have a positive set probability $w(C)>0$ for all $C \in \mathcal{C}$.
    ↪ Without the assumption that $w(C)>0$ for all $C \in \mathcal{C}$, the worst case error of any test of IIA would be trivially large, since it has no
    ↪ chance of measuring a violation of IIA within a set $C$ that it can not observe.
```

Lean-elaborated paper signature

```
field mass:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → F.Observation → ℝ

field nonneg:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (self : SeshadriUgander2020IIATesting.ChoiceSystem F)
  (o : F.Observation), 0 ≤ self.mass o

field sum_one:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (self : SeshadriUgander2020IIATesting.ChoiceSystem F),
  ∑ o : F.Observation, self.mass o = 1
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation

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

Reason: The source choice system is a joint probability distribution over observed item/choice-set incidences. The Lean mass function, nonnegativity, and total-mass-one fields state exactly that model.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign
Source locator: cited publication, lines 419-456

Verbatim paper-source connection

For the first simple step, consider the testing problem:
\begin{align}
\label{eq:test2}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for } p = p_0 \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta,
\end{cases}
\end{align}
where p_0 is the uniform distribution, $p_{0,(x,C)} = 1/d, \forall (x,C)$ (recalling that each discete distribution is d-dimensional, where d is the sum of the cardinalities of the choice sets in a given $\mathcal C$). Since $p_0 \in \mathcal{P}_0$, the problem is simpler than the original problem, and any lower bound on the performance of this problem carries over to the original problem of interest.

Now consider a discrete distribution $\qbe$ that is a perturbation of the uniform distribution of the form:
\begin{align*}
\qbe((x,C)) = \frac{1}{d} + \frac{\epsilon b_{(x,C)}}{d},
\end{align*}
\begin{align*}
\epsilon \in [0,1] \
b \in \lbrace -1, 1 \rbrace^d,
\
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\text{ and }
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x.
\end{align*}

That is, b perturbs the uniform distribution such that there is no net translation over any set, or over all of the appearances of an item over sets. Our goal is to construct perturbations $\qbe$, selecting ϵ and b that land in $\Mdelta$, meaning they are at least δ away (in TV distance) from $\Piia$ but at the same time still correspond to valid probability distributions (i.e., within the d-simplex). We are starting our journey (in the direction of b) at the uniform distribution p_0, and we want to show that we always land at least δ away from {\it any} distribution in $\Piia$.

Let $\Bset$ be any collection of $M=|\Bset|$ vectors b satisfying the above conditions for the given collection $\mathcal C$. That is,
\begin{align}
\label{eq:bset_full}
\Bset \subseteq
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x
\right \}.
\end{align}
We define $\bar{q}_\epsilon = \frac{1}{M} \sum_{b \in \Bset} \qbe$ as the mixture distribution over the M distributions in $\Bset$. Samples from $\bar{q}_\epsilon$ would be drawn marginally, that is, some $\qbe$ would be chosen uniformly from the M possible distributions, and samples would then come from that $\qbe$. Consider then the testing problem:

Lean-elaborated paper signature

```
field sign:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F → F.Observation → ℝ

field sign_eq_one_or_neg_one:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (self : SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F)
  (o : F.Observation), self.sign o = 1 ∨ self.sign o = -1

field set_balance:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (self : SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F)
  (C : F.SetId), ∑ x : ↥(F.members C), self.sign ⟨C, x⟩ = 0

field item_balance:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (self : SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F)
  (x : F.Item), ∑ C : { C : F.SetId // x ∈ F.members C }, self.sign ⟨↑C, ⟨x, C.property⟩⟩ = 0
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation

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

Reason: The declaration formalizes source equation (4): {-1,1}-valued perturbation signs whose sums vanish within every choice set and at every item. The indexed finite sums are the direct incidence-level form of the source balance constraints.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
Source locator: cited publication, lines 489-523

Verbatim paper-source connection

```
\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$, we
↪ begin by examining the cycle decompositions of $G_\mathcal{C}$.
    We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we
    ↪ call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to
    ↪ select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of
    ↪ $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

    \begin{figure}[t]
        \centering
        \includegraphics[height=55mm]{iia_graph_illustration.pdf}
        \caption{
            A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The
            ↪ hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph
            ↪ $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two
            ↪ undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles
            ↪ are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other
            ↪ orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
        }
        \label{fig:gc}
        \vspace{-4mm}
    \end{figure}
    A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$,
    ↪ and an element $\sigma_i \in \sigma$ denotes the $i$th cycle in $\sigma$. We use $|\sigma|$ to denote the number of cycles in the
    ↪ decomposition $\sigma$, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write
    ↪ $|\sigma_i|=2k_i$, where $k_i$ is the number of item-nodes in the cycle.

    Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation
    ↪ follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with $n$ item-nodes and $m$ set-nodes, any
    ↪ cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two
    ↪ smaller edge-disjoint cycles.

    \xhdr{From cycle decomposition to perturbations}
    We generate a structured collection of perturbation vectors $b$ by generating many distinct Eulerian orientations of a given cycle decomposition
    ↪ $\sigma$. We can independently direct the edges in each of the        cycles in $\sigma$ in one of 2 directions (clockwise or counterclockwise),
    ↪ and all of these collections of directions correspond to valid $b$ vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation
    ↪ for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector $b$.

    We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors $b$ that orient a cycle decomposition $\sigma$, refining our generic
    ↪ definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
    Here $(x,C)$ indexes the elements of the vector $b$ and we use $\ell$ to index sequential edges in a cycle $\sigma_i$. For each perturbation vector
    ↪ $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) =
    ↪ \frac{d}{|\sigma|}$, the average cycle length.
```

Lean-elaborated paper signature

```
field Cycle:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F → Type

field instFintypeCycle:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) → Fintype self.Cycle

field instDecidableEqCycle:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) → DecidableEq self.Cycle

field length:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) → self.Cycle → ℕ

field length_pos:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame}
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) (i : self.Cycle), 0 < self.length i

field edgeEquiv:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) →
    (i : self.Cycle) × Fin (self.length i) ≃ F.Observation
```

```
field base:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F → F.Observation → ℝ

field base_eq_one_or_neg_one:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame}
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) (o : F.Observation),
  self.base o = 1 ∨ self.base o = -1

field set_balance:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame}
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) (a : self.Cycle → Bool) (C : F.SetId),
  ∑ x : ↥(F.members C),
    self.base ⟨C, x⟩ * SeshadriUgander2020IIATesting.Rademacher.sign (a (self.edgeEquiv.symm ⟨C, x⟩).fst) =
    0

field item_balance:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame}
  (self : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) (a : self.Cycle → Bool) (x : F.Item),
  ∑ C : { C : F.SetId // x ∈ F.members C },
    self.base (↑C, ⟨x, C.property⟩) *
      SeshadriUgander2020IIATesting.Rademacher.sign (a (self.edgeEquiv.symm (↑C, ⟨x, C.property⟩)).fst) =
    0
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.Rademacher.sign

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

Reason: The source decomposes comparison-incidence edges into edge-disjoint simple cycles. The finite cycle index, positive lengths, and edge equivalence make coverage and disjointness explicit, while the alternating balance data records the source construction used for perturbations without changing the modeled decomposition.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.AlternatingCycleWitness

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.AlternatingCycleWitness
Source locator: cited publication, lines 489-523

Verbatim paper-source connection

\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$, we begin by examining the cycle decompositions of $G_\mathcal{C}$.
We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

\begin{figure}[t]
\centering
\includegraphics[height=55mm]{iia_graph_illustration.pdf}
\caption{
A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
}
\label{fig:gc}
\vspace{-4mm}
\end{figure}
A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$, and an element $\sigma_i \in \sigma$ denotes the ith cycle in σ. We use $|\sigma|$ to denote the number of cycles in the decomposition σ, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write $|\sigma_i|=2k_i$, where k_i is the number of item-nodes in the cycle.

Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with n item-nodes and m set-nodes, any cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two smaller edge-disjoint cycles.

\xhdr{From cycle decomposition to perturbations}
We generate a structured collection of perturbation vectors b by generating many distinct Eulerian orientations of a given cycle decomposition σ. We can independently direct the edges in each of the cycles in σ in one of 2 directions (clockwise or counterclockwise), and all of these collections of directions correspond to valid b vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector b.

We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors b that orient a cycle decomposition σ, refining our generic definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
Here (x,C) indexes the elements of the vector b and we use ℓ to index sequential edges in a cycle σ_i. For each perturbation vector $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) = \frac{d}{|\sigma|}$, the average cycle length.

Lean-elaborated paper signature

```
field Vertex:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} → D.AlternatingCycleWitness → D.Cycle → Type

field instFintypeVertex:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} →
    (self : D.AlternatingCycleWitness) → (i : D.Cycle) → Fintype (self.Vertex i)

field instDecidableEqVertex:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} →
    (self : D.AlternatingCycleWitness) → (i : D.Cycle) → DecidableEq (self.Vertex i)

field successor:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} →
    (self : D.AlternatingCycleWitness) → (i : D.Cycle) → Equiv.Perm (self.Vertex i)

field setAt:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} →
    (self : D.AlternatingCycleWitness) → (i : D.Cycle) → self.Vertex i → F.SetId
```

```
field itemAt:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} →
    (self : D.AlternatingCycleWitness) → (i : D.Cycle) → self.Vertex i → F.Item

field current_mem:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F}
  (self : D.AlternatingCycleWitness) (i : D.Cycle) (v : self.Vertex i), self.itemAt i v ∈ F.members (self.setAt i v)

field successor_mem:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F}
  (self : D.AlternatingCycleWitness) (i : D.Cycle) (v : self.Vertex i),
  self.itemAt i ((self.successor i) v) ∈ F.members (self.setAt i v)

field edgeParamEquiv:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F} →
    (self : D.AlternatingCycleWitness) → (i : D.Cycle) → Fin (D.length i) ≃ self.Vertex i × Bool

field false_edge:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F}
  (self : D.AlternatingCycleWitness) (i : D.Cycle) (v : self.Vertex i),
  D.edgeEquiv ⟨i, (self.edgeParamEquiv i).symm (v, false)⟩ = ⟨self.setAt i v, ⟨self.itemAt i v, self.current_mem i v⟩⟩

field true_edge:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F}
  (self : D.AlternatingCycleWitness) (i : D.Cycle) (v : self.Vertex i),
  D.edgeEquiv ⟨i, (self.edgeParamEquiv i).symm (v, true)⟩ =
    ⟨self.setAt i v, ⟨self.itemAt i ((self.successor i) v), self.successor_mem i v⟩⟩

field base_false:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F}
  (self : D.AlternatingCycleWitness) (i : D.Cycle) (v : self.Vertex i),
  D.base (D.edgeEquiv ⟨i, (self.edgeParamEquiv i).symm (v, false)⟩) = 1

field base_true:
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F}
  (self : D.AlternatingCycleWitness) (i : D.Cycle) (v : self.Vertex i),
  D.base (D.edgeEquiv ⟨i, (self.edgeParamEquiv i).symm (v, true)⟩) = -1
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition

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

Reason: The source lower-bound construction uses alternating traversals of its cycle decomposition to define signs on incidence edges. This witness is an explicit finite encoding of that traversal, its two alternating edges, and their signs; it introduces no additional source claim.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean
Source locator: cited publication, lines 489-523

Verbatim paper-source connection

\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$, we begin by examining the cycle decompositions of $G_\mathcal{C}$.
We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

\begin{figure}[t]
\centering
\includegraphics[height=55mm]{iia_graph_illustration.pdf}
\caption{
A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
}
\label{fig:gc}
\vspace{-4mm}
\end{figure}
A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$, and an element $\sigma_i \in \sigma$ denotes the ith cycle in σ. We use $|\sigma|$ to denote the number of cycles in the decomposition σ, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write $|\sigma_i|=2k_i$, where k_i is the number of item-nodes in the cycle.

Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with n item-nodes and m set-nodes, any cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two smaller edge-disjoint cycles.

\xhdr{From cycle decomposition to perturbations}
We generate a structured collection of perturbation vectors b by generating many distinct Eulerian orientations of a given cycle decomposition σ. We can independently direct the edges in each of the cycles in σ in one of 2 directions (clockwise or counterclockwise), and all of these collections of directions correspond to valid b vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector b.

We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors b that orient a cycle decomposition σ, refining our generic definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
Here (x,C) indexes the elements of the vector b and we use ℓ to index sequential edges in a cycle σ_i. For each perturbation vector $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) = \frac{d}{|\sigma|}$, the average cycle length.

Lean-expanded paper semantic target

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame}
(D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F) =>
↑F.incidenceCount / ↑(Fintype.card D.Cycle)

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition

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

Reason: The target is the source definition mu(sigma)=d/|sigma|, using the formal incidence count for d and the finite cycle index for |sigma|. Its coercions only express the same real-valued quotient.

Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.SatisfiesIIA

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.SatisfiesIIA
Source locator: cited publication, lines 248-260

Verbatim paper-source connection

```
\subsection{Choice systems and IIA}
    The Independence of Irrelevant Alternatives (IIA) assumption constrains the probabilities $P_{x,C}$ to satisfy the following ratio for any $C \subset
    ↪  \mathcal{X}$:
    \begin{align*}
    \dfrac{P_{x,\lbrace x, y \rbrace \cup C}}{P_{y,\lbrace x, y \rbrace \cup C}} = \dfrac{P_{x,\lbrace x, y \rbrace}}{P_{y,\lbrace x, y \rbrace}}.
    \end{align*}
    As derived by Luce in \cite{luce1959individual}, the IIA assumption consequently admits what Luce called a ratio representation,
    \begin{align*}
    P_{x,C} = \frac{\gamma_x}{\sum_{z \in C} \gamma_z},
    \end{align*}
    for some $\gamma \in \mathbb{R}_+^n$. Clearly, $\gamma$ is scale invariant. Setting a scale such that $\sum_{z \in \mathcal{X}} \gamma_z = 1$
    ↪  gives the interpretation that $\gamma_z = P_{z, \mathcal{X}}$.
    That is, this interpretation says that an IIA choice system can be described by just knowing $w(C)$, $\forall  C \in \mathcal{C}$, and
    ↪  $P_{x,\mathcal{X}}$, $\forall x \in \mathcal X$.

    We let $\Piia \subset \Pall$ denote the collection of all choice systems that satisfy IIA. Essentially, then, where $\Pall$ is the space of all
    ↪  $d$-dimensional discrete distributions, $\Piia$ is a family of (non-linearly) constrained $d$-dimensional discrete distributions. We emphasize that
    ↪  $\Piia$ is non-convex, and that the convex hull of $\Piia$ is dense in $\Pall$; the latter fact we show in the appendix (Fact~\ref{cvx_hull_dense}).
```

Lean-expanded paper semantic target

type:
```
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → Prop
```

value:
```
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  ∃ (γ : F.Item → ℝ),
    (∀ (x : F.Item), 0 < γ x) ∧
      ∀ (C : F.SetId) (x : ↥(F.members C)), q.mass (C, x) = q.setMass C * γ ↑x / ∑ z ∈ F.members C, γ z
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.setMass

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

Reason: The source IIA condition equates choice ratios for every pair of items across all sets containing that pair. The target states that same ratio condition with positive set masses made explicit for the logarithmic/probabilistic domain.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.totalVariation

Paper-local declaration:

SeshadriUgander2020IIATesting.ChoiceSystem.totalVariation
Source locator: cited publication, lines 262-315

Verbatim paper-source connection

\subsection{Separation from IIA}

Let $\Mdelta \subset \Pall$ denote the set of non-IIA choice systems that are δ-separated, in total variation distance, from the set of IIA choice systems $\Piia$, for a given collection of choice sets $\mathcal C$:
\begin{align}
\Mdelta = \{q : q \in \Pall, \ \inf_{p \in \Piia} \tv{q-p} \geq \delta\}.
\end{align}
Intuitively, δ describes the size of the \textit{indifference zone} \cite{lehmann2005testing} between the null and the alternative hypothesis, beyond which false acceptance becomes a serious error and ought to be limited. Since IIA and ``not IIA'' denote two contiguous sets, the separation makes the division of parameters sharp, and helps define the following testing problem, the central problem of this work:
\begin{align}
\label{eq:iiatest}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for some unknown } p \in \Piia \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta.
\end{cases}
\end{align}

Given the hypotheses in ~\eqref{eq:iiatest}, we define a test ϕ as a map, $\phi : \{ (x_1,C_1),...,(x_N,C_N) \} \mapsto \{0,1\}$ from the dataset to a decision to reject the null H_0. Our choice of total variation to define the separation of the testing problem is motivated by the distance metric's natural interpretation as describing the maximal gap in probability between two distributions over all possible events. Moreover, the choice is aligned with many other recent analyses of testing for discrete distributions~\cite{paninski2008coincidence,wei2016sharp,valiant2017automatic} and thus enables a direct comparison of the difficulty of testing IIA with the difficulty of testing other properties such as identity or monotonicity.

Several other measures of distance for the separation, e.g., Hellinger distance or Kullback-Liebler divergence, may also be reasonable~\cite{acharya2015optimal,daskalakis2018distribution}. The Hellinger distance metric is closely related to total variation distance and has been shown in certain testing scenarios to produce rates identical to those resulting from total variation distance~\cite{daskalakis2018distribution}. We conjecture this is also the case for the IIA testing problem, and leave such an exploration for future work. The Kullback-Liebler divergence is a weaker measure of separation; although it has been shown to lead to trivial (infinite) minimax risk when used in general problems of identity testing in discrete distributions, this concern does not arise for the case for IIA testing\footnote{Whereas in identity testing, two distributions can be arbitrarily close while having infinite KL divergence, the KL divergence to the closest point within IIA is always upper bounded by a finite quantity strictly decreasing with total variation distance.}. The measure is, however, unusual and difficult to interpret. As stated before, consider the d-dimensional uniform distribution and distributions ϵ far away in total variation distance; the KL divergence is $\mathcal{O}(d)$ larger than the TV distance when the distributional difference lies within a constant number of entries (as opposed to being spread evenly across all of the entries). For these reasons, we focus on total variation distance in this work.

\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with n nodes for each item in $\mathcal{X}$ and m nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in C. This bipartite graph can also be thought of as representing a hypergraph with n nodes, one for each item in $\mathcal{X}$, and m hyperedges, one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain d edges. As a result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item i to every item j.

This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice systems are IIA, rendering a test for IIA meaningless.

We call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by σ. We define two functions of σ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms further in Section \ref{sec:bounding}.

For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes the requirement of even set sizes and even item appearances.

\section{Lower bound on the minimax risk}
\label{sec:lowerbounds}

Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the IIA property against the alternative of an arbitrary choice system that is δ-separated from IIA. We seek to lower bound the minimax risk of this testing problem. The maximum risk of any test ϕ is a useful quantity that demonstrates how poor the test could be at finding IIA violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case scenario \cite{balakrishnan2018hypothesis}.

A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number of samples N, the separation δ between the null and the alternative, and problem parameters. It is a statement about the fundamental possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

\subsection{Minimax risk preliminaries}
The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
\begin{align*}
R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) = 0),
\end{align*}
where the $\inf$ is taken over any test ϕ. The testing radius $\delta_N(\Piia)$ is then be defined as
\begin{align*}
\delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
\end{align*}
where a is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone that allows hypotheses to be uniformly distinguishable with probability $1-a$.
Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
\begin{align*}
N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
\end{align*}

For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-α tests, with the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.

Lean-expanded paper semantic target

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem F → SeshadriUgander2020IIATesting.ChoiceSystem F → ℝ

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q r : SeshadriUgander2020IIATesting.ChoiceSystem F) =>
  1 / 2 * ∑ o : F.Observation, |q.mass o - r.mass o|
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem

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

Reason: The declaration is the source total-variation distance on the finite choice-system sample space: one half of the sum of absolute mass differences over all observed incidences.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.SeparatedFromIIA

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.SeparatedFromIIA
Source locator: cited publication, lines 262-315

Verbatim paper-source connection

\subsection{Separation from IIA}

Let $\Mdelta \subset \Pall$ denote the set of non-IIA choice systems that are δ-separated, in total variation distance, from the set of IIA choice systems $\Piia$, for a given collection of choice sets $\mathcal C$:
\begin{align}
\Mdelta = \{q : q \in \Pall, \ \inf_{p \in \Piia} \tv{q-p} \geq \delta\}.
\end{align}
Intuitively, δ describes the size of the \textit{indifference zone} \cite{lehmann2005testing} between the null and the alternative hypothesis, beyond which false acceptance becomes a serious error and ought to be limited. Since IIA and ``not IIA'' denote two contiguous sets, the separation makes the division of parameters sharp, and helps define the following testing problem, the central problem of this work:
\begin{align}
\label{eq:iiatest}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for some unknown } p \in \Piia \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta.
\end{cases}
\end{align}

Given the hypotheses in ~\eqref{eq:iiatest}, we define a test ϕ as a map, $\phi : \{ (x_1,C_1),...,(x_N,C_N) \} \mapsto \{0,1\}$ from the dataset to a decision to reject the null H_0. Our choice of total variation to define the separation of the testing problem is motivated by the distance metric's natural interpretation as describing the maximal gap in probability between two distributions over all possible events. Moreover, the choice is aligned with many other recent analyses of testing for discrete distributions~\cite{paninski2008coincidence,wei2016sharp,valiant2017automatic} and thus enables a direct comparison of the difficulty of testing IIA with the difficulty of testing other properties such as identity or monotonicity.

Several other measures of distance for the separation, e.g., Hellinger distance or Kullback-Liebler divergence, may also be reasonable~\cite{acharya2015optimal,daskalakis2018distribution}. The Hellinger distance metric is closely related to total variation distance and has been shown in certain testing scenarios to produce rates identical to those resulting from total variation distance~\cite{daskalakis2018distribution}. We conjecture this is also the case for the IIA testing problem, and leave such an exploration for future work. The Kullback-Liebler divergence is a weaker measure of separation; although it has been shown to lead to trivial (infinite) minimax risk when used in general problems of identity testing in discrete distributions, this concern does not arise for the case for IIA testing\footnote{Whereas in identity testing, two distributions can be arbitrarily close while having infinite KL divergence, the KL divergence to the closest point within IIA is always upper bounded by a finite quantity strictly decreasing with total variation distance.}. The measure is, however, unusual and difficult to interpret. As stated before, consider the d-dimensional uniform distribution and distributions ϵ far away in total variation distance; the KL divergence is $\mathcal{O}(d)$ larger than the TV distance when the distributional difference lies within a constant number of entries (as opposed to being spread evenly across all of the entries). For these reasons, we focus on total variation distance in this work.

\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with n nodes for each item in $\mathcal{X}$ and m nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in C. This bipartite graph can also be thought of as representing a hypergraph with n nodes, one for each item in $\mathcal{X}$, and m hyperedges, one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain d edges. As a result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item i to every item j.

This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice systems are IIA, rendering a test for IIA meaningless.

We call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by σ. We define two functions of σ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms further in Section \ref{sec:bounding}.

For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes the requirement of even set sizes and even item appearances.

\section{Lower bound on the minimax risk}
\label{sec:lowerbounds}

Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the IIA property against the alternative of an arbitrary choice system that is δ-separated from IIA. We seek to lower bound the minimax risk of this testing problem. The maximum risk of any test ϕ is a useful quantity that demonstrates how poor the test could be at finding IIA violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case scenario \cite{balakrishnan2018hypothesis}.

A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number of samples N, the separation δ between the null and the alternative, and problem parameters. It is a statement about the fundamental possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

\subsection{Minimax risk preliminaries}
The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
\begin{align*}
R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) = 0),
\end{align*}
where the $\inf$ is taken over any test ϕ. The testing radius $\delta_N(\Piia)$ is then be defined as
\begin{align*}
\delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
\end{align*}
where a is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone that allows hypotheses to be uniformly distinguishable with probability $1-a$.
Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
\begin{align*}
N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
\end{align*}

For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-α tests, with the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.

Lean-expanded paper semantic target

type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → SeshadriUgander2020IIATesting.ChoiceSystem F → ℝ → Prop

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) (δ : ℝ) =>
  ∀ (p : SeshadriUgander2020IIATesting.ChoiceSystem F), p.SatisfiesIIA → δ ≤ q.totalVariation p

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.SatisfiesIIA
- SeshadriUgander2020IIATesting.ChoiceSystem.totalVariation

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

Reason: The source alternative consists of choice systems at least delta away from every IIA choice system in total variation. The target is exactly that universally quantified separation predicate.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.FiniteDistribution.Test

Paper-local declaration:
SeshadriUgander2020IIATesting.FiniteDistribution.Test
Source locator: cited publication, lines 262-315

Verbatim paper-source connection

\subsection{Separation from IIA}

Let $\Mdelta \subset \Pall$ denote the set of non-IIA choice systems that are δ-separated, in total variation distance, from the set of IIA choice systems $\Piia$, for a given collection of choice sets $\mathcal C$:
\begin{align}
\Mdelta = \{q : q \in \Pall, \ \inf_{p \in \Piia} \tv{q-p} \geq \delta\}.
\end{align}
Intuitively, δ describes the size of the \textit{indifference zone} \cite{lehmann2005testing} between the null and the alternative hypothesis, beyond which false acceptance becomes a serious error and ought to be limited. Since IIA and ``not IIA'' denote two contiguous sets, the separation makes the division of parameters sharp, and helps define the following testing problem, the central problem of this work:
\begin{align}
\label{eq:iiatest}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for some unknown } p \in \Piia \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta.
\end{cases}
\end{align}

Given the hypotheses in ~\eqref{eq:iiatest}, we define a test ϕ as a map, $\phi : \{ (x_1,C_1),...,(x_N,C_N) \} \mapsto \{0,1\}$ from the dataset to a decision to reject the null H_0. Our choice of total variation to define the separation of the testing problem is motivated by the distance metric's natural interpretation as describing the maximal gap in probability between two distributions over all possible events. Moreover, the choice is aligned with many other recent analyses of testing for discrete distributions~\cite{paninski2008coincidence,wei2016sharp,valiant2017automatic} and thus enables a direct comparison of the difficulty of testing IIA with the difficulty of testing other properties such as identity or monotonicity.

Several other measures of distance for the separation, e.g., Hellinger distance or Kullback-Liebler divergence, may also be reasonable~\cite{acharya2015optimal,daskalakis2018distribution}. The Hellinger distance metric is closely related to total variation distance and has been shown in certain testing scenarios to produce rates identical to those resulting from total variation distance~\cite{daskalakis2018distribution}. We conjecture this is also the case for the IIA testing problem, and leave such an exploration for future work. The Kullback-Liebler divergence is a weaker measure of separation; although it has been shown to lead to trivial (infinite) minimax risk when used in general problems of identity testing in discrete distributions, this concern does not arise for the case for IIA testing\footnote{Whereas in identity testing, two distributions can be arbitrarily close while having infinite KL divergence, the KL divergence to the closest point within IIA is always upper bounded by a finite quantity strictly decreasing with total variation distance.}. The measure is, however, unusual and difficult to interpret. As stated before, consider the d-dimensional uniform distribution and distributions ϵ far away in total variation distance; the KL divergence is $\mathcal{O}(d)$ larger than the TV distance when the distributional difference lies within a constant number of entries (as opposed to being spread evenly across all of the entries). For these reasons, we focus on total variation distance in this work.

\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with n nodes for each item in $\mathcal{X}$ and m nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in C. This bipartite graph can also be thought of as representing a hypergraph with n nodes, one for each item in $\mathcal{X}$, and m hyperedges, one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain d edges. As a result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item i to every item j.

This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice systems are IIA, rendering a test for IIA meaningless.

We call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by σ. We define two functions of σ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms further in Section \ref{sec:bounding}.

For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes the requirement of even set sizes and even item appearances.

\section{Lower bound on the minimax risk}
\label{sec:lowerbounds}

Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the IIA property against the alternative of an arbitrary choice system that is δ-separated from IIA. We seek to lower bound the minimax risk of this testing problem. The maximum risk of any test ϕ is a useful quantity that demonstrates how poor the test could be at finding IIA violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case scenario \cite{balakrishnan2018hypothesis}.

A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number of samples N, the separation δ between the null and the alternative, and problem parameters. It is a statement about the fundamental possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

\subsection{Minimax risk preliminaries}
The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
\begin{align*}
R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) = 0),
\end{align*}
where the $\inf$ is taken over any test ϕ. The testing radius $\delta_N(\Piia)$ is then be defined as
\begin{align*}
\delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
\end{align*}
where a is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone that allows hypotheses to be uniformly distinguishable with probability $1-a$.
Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
\begin{align*}
N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
\end{align*}

For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-α tests, with the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.

Lean-expanded paper semantic target

type:
Type → Type

value:
fun (α : Type) => α → Bool

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

Reason: The source treats a test as a rule applied to N independent observations from a finite choice system. The finite-distribution test type is exactly that Boolean-valued decision rule on the product sample space.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.FiniteDistribution.binaryError

Paper-local declaration:
SeshadriUgander2020IIATesting.FiniteDistribution.binaryError
Source locator: cited publication, lines 262-315

Verbatim paper-source connection

\subsection{Separation from IIA}

Let $\Mdelta \subset \Pall$ denote the set of non-IIA choice systems that are δ-separated, in total variation distance, from the set of IIA choice systems $\Piia$, for a given collection of choice sets $\mathcal C$:
\begin{align}
\Mdelta = \{q : q \in \Pall, \ \inf_{p \in \Piia} \tv{q-p} \geq \delta\}.
\end{align}
Intuitively, δ describes the size of the \textit{indifference zone} \cite{lehmann2005testing} between the null and the alternative hypothesis, beyond which false acceptance becomes a serious error and ought to be limited. Since IIA and ``not IIA'' denote two contiguous sets, the separation makes the division of parameters sharp, and helps define the following testing problem, the central problem of this work:
\begin{align}
\label{eq:iiatest}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for some unknown } p \in \Piia \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta.
\end{cases}
\end{align}

Given the hypotheses in ~\eqref{eq:iiatest}, we define a test ϕ as a map, $\phi : \{ (x_1,C_1),...,(x_N,C_N) \} \mapsto \{0,1\}$ from the dataset to a decision to reject the null H_0. Our choice of total variation to define the separation of the testing problem is motivated by the distance metric's natural interpretation as describing the maximal gap in probability between two distributions over all possible events. Moreover, the choice is aligned with many other recent analyses of testing for discrete distributions~\cite{paninski2008coincidence,wei2016sharp,valiant2017automatic} and thus enables a direct comparison of the difficulty of testing IIA with the difficulty of testing other properties such as identity or monotonicity.

Several other measures of distance for the separation, e.g., Hellinger distance or Kullback-Liebler divergence, may also be reasonable~\cite{acharya2015optimal,daskalakis2018distribution}. The Hellinger distance metric is closely related to total variation distance and has been shown in certain testing scenarios to produce rates identical to those resulting from total variation distance~\cite{daskalakis2018distribution}. We conjecture this is also the case for the IIA testing problem, and leave such an exploration for future work. The Kullback-Liebler divergence is a weaker measure of separation; although it has been shown to lead to trivial (infinite) minimax risk when used in general problems of identity testing in discrete distributions, this concern does not arise for the case for IIA testing\footnote{Whereas in identity testing, two distributions can be arbitrarily close while having infinite KL divergence, the KL divergence to the closest point within IIA is always upper bounded by a finite quantity strictly decreasing with total variation distance.}. The measure is, however, unusual and difficult to interpret. As stated before, consider the d-dimensional uniform distribution and distributions ϵ far away in total variation distance; the KL divergence is $\mathcal{O}(d)$ larger than the TV distance when the distributional difference lies within a constant number of entries (as opposed to being spread evenly across all of the entries). For these reasons, we focus on total variation distance in this work.

\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with n nodes for each item in $\mathcal{X}$ and m nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in C. This bipartite graph can also be thought of as representing a hypergraph with n nodes, one for each item in $\mathcal{X}$, and m hyperedges, one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain d edges. As a result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item i to every item j.

This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice systems are IIA, rendering a test for IIA meaningless.

We call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by σ. We define two functions of σ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms further in Section \ref{sec:bounding}.

For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes the requirement of even set sizes and even item appearances.

\section{Lower bound on the minimax risk}
\label{sec:lowerbounds}

Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the IIA property against the alternative of an arbitrary choice system that is δ-separated from IIA. We seek to lower bound the minimax risk of this testing problem. The maximum risk of any test ϕ is a useful quantity that demonstrates how poor the test could be at finding IIA violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case scenario \cite{balakrishnan2018hypothesis}.

A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number of samples N, the separation δ between the null and the alternative, and problem parameters. It is a statement about the fundamental possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

\subsection{Minimax risk preliminaries}
The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
\begin{align*}
R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) = 0),
\end{align*}
where the $\inf$ is taken over any test ϕ. The testing radius $\delta_N(\Piia)$ is then be defined as
\begin{align*}
\delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
\end{align*}
where a is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone that allows hypotheses to be uniformly distinguishable with probability $1-a$.
Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
\begin{align*}
N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
\end{align*}

For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-α tests, with the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.

Lean-expanded paper semantic target

```
type:
{α : Type} →
  [inst : Fintype α] →
    [DecidableEq α] →
      SeshadriUgander2020IIATesting.FiniteDistribution α →
        SeshadriUgander2020IIATesting.FiniteDistribution α → SeshadriUgander2020IIATesting.FiniteDistribution.Test α → ℝ

value:
fun {α : Type} [Fintype α] [DecidableEq α] (p q : SeshadriUgander2020IIATesting.FiniteDistribution α)
    (φ : SeshadriUgander2020IIATesting.FiniteDistribution.Test α) =>
  1 / 2 * ∑ x ∈ SeshadriUgander2020IIATesting.FiniteDistribution.rejectionSet φ, p.mass x +
    1 / 2 * ∑ x ∈ (SeshadriUgander2020IIATesting.FiniteDistribution.rejectionSet φ)ᶜ, q.mass x
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.FiniteDistribution
- SeshadriUgander2020IIATesting.FiniteDistribution.Test
- SeshadriUgander2020IIATesting.FiniteDistribution.rejectionSet

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

Reason: The target is the source binary testing error: the half-weighted sum of rejection probability under the null distribution and acceptance probability under the alternative. It is the error measure used in the minimax-risk definition.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest
Source locator: cited publication, lines 262-315

Verbatim paper-source connection

```
\subsection{Separation from IIA}

    Let $\Mdelta \subset \Pall$ denote the set of non-IIA choice systems that are $\delta$-separated, in total variation distance, from the set of IIA choice
    ↪ systems $\Piia$, for a given collection of choice sets $\mathcal C$:
\begin{align}
\Mdelta = \{q : q \in \Pall, \ \inf_{p \in \Piia} \tv{q-p} \geq \delta\}.
\end{align}
Intuitively, $\delta$ describes the size of the \textit{indifference zone} \cite{lehmann2005testing} between the null and the alternative hypothesis,
    ↪ beyond which false acceptance becomes a serious error and ought to be limited. Since IIA and ``not IIA'' denote two contiguous sets, the
    ↪ separation makes the division of parameters sharp, and helps define the following testing problem, the central problem of this work:
\begin{align}
\label{eq:iiatest}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for some unknown } p \in \Piia \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta.
\end{cases}
\end{align}

    Given the hypotheses in ~\eqref{eq:iiatest}, we define a test $\phi$ as a map, $\phi : \{ (x_1,C_1),...,(x_N,C_N) \} \mapsto \{0,1\}$ from the dataset
    ↪ to a decision to reject the null $H_0$. Our choice of total variation to define the separation of the testing problem is motivated by the distance
    ↪ metric's natural interpretation as describing the maximal gap in probability between two distributions over all possible events. Moreover, the
    ↪ choice is aligned with many other recent analyses of testing for discrete
    ↪ distributions~\cite{paninski2008coincidence,wei2016sharp,valiant2017automatic} and thus enables a direct comparison of the difficulty of
    ↪ testing IIA with the difficulty of testing other properties such as identity or monotonicity.

    Several other measures of distance for the separation, e.g., Hellinger distance or Kullback-Liebler divergence, may also be
    ↪ reasonable~\cite{acharya2015optimal,daskalakis2018distribution}. The Hellinger distance metric is closely related to total variation distance
    ↪ and has been shown in certain testing scenarios to produce rates identical to those resulting from total variation
    ↪ distance~\cite{daskalakis2018distribution}. We conjecture this is also the case for the IIA testing problem, and leave such an exploration for
    ↪ future work. The Kullback-Liebler divergence is a weaker measure of separation; although it has been shown to lead to trivial (infinite) minimax
    ↪ risk when used in general problems of identity testing in discrete distributions, this concern does not arise for the case for IIA
    ↪ testing\footnote{Whereas in identity testing, two distributions can be arbitrarily close while having infinite KL divergence, the KL divergence to
    ↪ the closest point within IIA is always upper bounded by a finite quantity strictly decreasing with total variation distance.}. The measure is,
    ↪ however, unusual and difficult to interpret. As stated before, consider the $d$-dimensional uniform distribution and distributions $\epsilon$ far
    ↪ away in total variation distance; the KL divergence is $\mathcal{O}(d)$ larger than the TV distance when the distributional difference lies within
    ↪ a constant number of entries (as opposed to being spread evenly across all of the entries). For these reasons, we focus on total variation
    ↪ distance in this work.

    \subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
    To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with $n$ nodes for
    ↪ each item in $\mathcal{X}$ and $m$ nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes
    ↪ to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in $C$. This
    ↪ bipartite graph can also be thought of as representing a hypergraph with $n$ nodes, one for each item in $\mathcal{X}$, and $m$ hyperedges,
    ↪ one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain $d$ edges. As a
    ↪ result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the
    ↪ study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

    Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item $i$ to
    ↪ every item $j$.

    This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by
    ↪ Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq
    ↪ n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and
    ↪ sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we
    ↪ prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we
    ↪ shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice
    ↪ systems are IIA, rendering a test for IIA meaningless.

    We call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by
    ↪ $\sigma$. We define two functions of $\sigma$ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle
    ↪ length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion
    ↪ index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to
    ↪ a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms
    ↪ further in Section \ref{sec:bounding}.

    For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$
    ↪ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in
    ↪ $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a
    ↪ consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
    We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do
    ↪ not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes
    ↪ the requirement of even set sizes and even item appearances.

    \section{Lower bound on the minimax risk}
    \label{sec:lowerbounds}

    Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the
    ↪ IIA property against the alternative of an arbitrary choice system that is $\delta$-separated from IIA. We seek to lower bound the minimax risk of
    ↪ this testing problem. The maximum risk of any test $\phi$ is a useful quantity that demonstrates how poor the test could be at finding IIA
    ↪ violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of
    ↪ violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case
    ↪ scenario \cite{balakrishnan2018hypothesis}.
```

A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number of samples N, the separation δ between the null and the alternative, and problem parameters. It is a statement about the fundamental possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

\subsection{Minimax risk preliminaries}
The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
\begin{align*}
R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) = 0),
\end{align*}
where the $\inf$ is taken over any test ϕ. The testing radius $\delta_N(\Piia)$ is then be defined as
\begin{align*}
\delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
\end{align*}
where a is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone that allows hypotheses to be uniformly distinguishable with probability $1-a$.
Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
\begin{align*}
N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
\end{align*}

For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-α tests, with the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.

Lean-expanded paper semantic target

type:
```
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → ℕ → ℝ → Prop
```

value:
```
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (N : ℕ) (δ : ℝ) =>
  ∃ (φ : SeshadriUgander2020IIATesting.FiniteDistribution.Test (Fin N → F.Observation)),
    ∀ (q : SeshadriUgander2020IIATesting.ChoiceSystem F),
      q.SeparatedFromIIA δ →
        ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N).binaryError
            (q.asFiniteDistribution.product N) φ ≤
          1 / 4
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instDecidableEqObservation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.SeparatedFromIIA
- SeshadriUgander2020IIATesting.ChoiceSystem.asFiniteDistribution
- SeshadriUgander2020IIATesting.ChoiceSystem.uniform
- SeshadriUgander2020IIATesting.FiniteDistribution.Test
- SeshadriUgander2020IIATesting.FiniteDistribution.binaryError
- SeshadriUgander2020IIATesting.FiniteDistribution.product

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

Reason: The source testing model measures the worst-case error of a test between IIA systems and systems delta-separated from IIA. The target packages the quarter-error threshold used by the source lower-bound and sample-complexity conclusions without changing the null or alternative.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound
Source locator: cited publication, lines 262-315

Verbatim paper-source connection

\subsection{Separation from IIA}

Let $\Mdelta \subset \Pall$ denote the set of non-IIA choice systems that are δ-separated, in total variation distance, from the set of IIA choice systems $\Piia$, for a given collection of choice sets $\mathcal C$:
\begin{align}
\Mdelta = \{q : q \in \Pall, \ \inf_{p \in \Piia} \tv{q-p} \geq \delta\}.
\end{align}
Intuitively, δ describes the size of the \textit{indifference zone} \cite{lehmann2005testing} between the null and the alternative hypothesis, beyond which false acceptance becomes a serious error and ought to be limited. Since IIA and ``not IIA'' denote two contiguous sets, the separation makes the division of parameters sharp, and helps define the following testing problem, the central problem of this work:
\begin{align}
\label{eq:iiatest}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for some unknown } p \in \Piia \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta.
\end{cases}
\end{align}

Given the hypotheses in ~\eqref{eq:iiatest}, we define a test ϕ as a map, $\phi : \{ (x_1,C_1),...,(x_N,C_N) \} \mapsto \{0,1\}$ from the dataset to a decision to reject the null H_0. Our choice of total variation to define the separation of the testing problem is motivated by the distance metric's natural interpretation as describing the maximal gap in probability between two distributions over all possible events. Moreover, the choice is aligned with many other recent analyses of testing for discrete distributions~\cite{paninski2008coincidence,wei2016sharp,valiant2017automatic} and thus enables a direct comparison of the difficulty of testing IIA with the difficulty of testing other properties such as identity or monotonicity.

Several other measures of distance for the separation, e.g., Hellinger distance or Kullback-Liebler divergence, may also be reasonable~\cite{acharya2015optimal,daskalakis2018distribution}. The Hellinger distance metric is closely related to total variation distance and has been shown in certain testing scenarios to produce rates identical to those resulting from total variation distance~\cite{daskalakis2018distribution}. We conjecture this is also the case for the IIA testing problem, and leave such an exploration for future work. The Kullback-Liebler divergence is a weaker measure of separation; although it has been shown to lead to trivial (infinite) minimax risk when used in general problems of identity testing in discrete distributions, this concern does not arise for the case for IIA testing\footnote{Whereas in identity testing, two distributions can be arbitrarily close while having infinite KL divergence, the KL divergence to the closest point within IIA is always upper bounded by a finite quantity strictly decreasing with total variation distance.}. The measure is, however, unusual and difficult to interpret. As stated before, consider the d-dimensional uniform distribution and distributions ϵ far away in total variation distance; the KL divergence is $\mathcal{O}(d)$ larger than the TV distance when the distributional difference lies within a constant number of entries (as opposed to being spread evenly across all of the entries). For these reasons, we focus on total variation distance in this work.

\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with n nodes for each item in $\mathcal{X}$ and m nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in C. This bipartite graph can also be thought of as representing a hypergraph with n nodes, one for each item in $\mathcal{X}$, and m hyperedges, one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain d edges. As a result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item i to every item j.

This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice systems are IIA, rendering a test for IIA meaningless.

We call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by σ. We define two functions of σ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms further in Section \ref{sec:bounding}.

For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes the requirement of even set sizes and even item appearances.

\section{Lower bound on the minimax risk}
\label{sec:lowerbounds}

Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the IIA property against the alternative of an arbitrary choice system that is δ-separated from IIA. We seek to lower bound the minimax risk of this testing problem. The maximum risk of any test ϕ is a useful quantity that demonstrates how poor the test could be at finding IIA violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case scenario \cite{balakrishnan2018hypothesis}.

A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number of samples N, the separation δ between the null and the alternative, and problem parameters. It is a statement about the fundamental possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

\subsection{Minimax risk preliminaries}
The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
\begin{align*}
R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) = 0),
\end{align*}
where the $\inf$ is taken over any test ϕ. The testing radius $\delta_N(\Piia)$ is then be defined as
\begin{align*}
\delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
\end{align*}
where a is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone that allows hypotheses to be uniformly distinguishable with probability $1-a$.
Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
\begin{align*}
N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
\end{align*}

For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-α tests, with the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.

Lean-expanded paper semantic target

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} → ℕ → ℝ → ℝ → Prop

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (N : ℕ) (δ lower : ℝ) =>
  ∀ (φ : SeshadriUgander2020IIATesting.FiniteDistribution.Test (Fin N → F.Observation)),
    ∃ (q : SeshadriUgander2020IIATesting.ChoiceSystem F),
      q.SeparatedFromIIA δ ∧
        lower ≤
          ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N).binaryError
            (q.asFiniteDistribution.product N) φ
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instDecidableEqObservation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.SeparatedFromIIA
- SeshadriUgander2020IIATesting.ChoiceSystem.asFiniteDistribution
- SeshadriUgander2020IIATesting.ChoiceSystem.uniform
- SeshadriUgander2020IIATesting.FiniteDistribution.Test
- SeshadriUgander2020IIATesting.FiniteDistribution.binaryError
- SeshadriUgander2020IIATesting.FiniteDistribution.product

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

Reason: The declaration gives the source product-sample minimax testing-risk lower-bound predicate: every N-sample test has the stated binary error against the IIA null and delta-separated alternatives. This is the formal test problem used by Theorem 1.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.ChoiceSystem.perturb

Paper-local declaration:
SeshadriUgander2020IIATesting.ChoiceSystem.perturb
Source locator: cited publication, lines 419-456

Verbatim paper-source connection

For the first simple step, consider the testing problem:
\begin{align}
\label{eq:test2}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for } p = p_0 \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta,
\end{cases}
\end{align}
where p_0 is the uniform distribution, $p_{0,(x,C)} = 1/d, \forall (x,C)$ (recalling that each discete distribution is d-dimensional, where d is the sum of the cardinalities of the choice sets in a given $\mathcal C$). Since $p_0 \in \mathcal{P}_0$, the problem is simpler than the original problem, and any lower bound on the performance of this problem carries over to the original problem of interest.

Now consider a discrete distribution $\qbe$ that is a perturbation of the uniform distribution of the form:
\begin{align*}
\qbe((x,C)) = \frac{1}{d} + \frac{\epsilon b_{(x,C)}}{d},
\end{align*}
\begin{align*}
\epsilon \in [0,1] \
b \in \lbrace -1, 1 \rbrace^d,
\
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\text{ and }
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x.
\end{align*}

That is, b perturbs the uniform distribution such that there is no net translation over any set, or over all of the appearances of an item over sets. Our goal is to construct perturbations $\qbe$, selecting ϵ and b that land in $\Mdelta$, meaning they are at least δ away (in TV distance) from $\Piia$ but at the same time still correspond to valid probability distributions (i.e., within the d-simplex). We are starting our journey (in the direction of b) at the uniform distribution p_0, and we want to show that we always land at least δ away from {\it any} distribution in $\Piia$.

Let $\Bset$ be any collection of $M=|\Bset|$ vectors b satisfying the above conditions for the given collection $\mathcal C$. That is,
\begin{align}
\label{eq:bset_full}
\Bset \subseteq
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x
\right \}.
\end{align}
We define $\bar{q}_\epsilon = \frac{1}{M} \sum_{b \in \Bset} \qbe$ as the mixture distribution over the M distributions in $\Bset$. Samples from $\bar{q}_\epsilon$ would be drawn marginally, that is, some $\qbe$ would be chosen uniformly from the M possible distributions, and samples would then come from that $\qbe$. Consider then the testing problem:

Lean-expanded paper semantic target

```
type:
{F : SeshadriUgander2020IIATesting.ChoiceFrame} →
  SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F →
    (ε : ℝ) → 0 ≤ ε → ε ≤ 1 → SeshadriUgander2020IIATesting.ChoiceSystem F

value:
fun {F : SeshadriUgander2020IIATesting.ChoiceFrame} (b : SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F)
    (ε : ℝ) (hε_nonneg : 0 ≤ ε) (hε_le_one : ε ≤ 1) =>
  { mass := fun (o : F.Observation) => (1 + ε * b.sign o) / ↑F.incidenceCount,
    nonneg := SeshadriUgander2020IIATesting.ChoiceSystem.perturb._proof_1 b ε hε_nonneg hε_le_one,
    sum_one := SeshadriUgander2020IIATesting.ChoiceSystem.perturb._proof_2 b ε }
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign

Recorded source-to-declaration screening **Verdict:** matches
Reason: The target is the source balanced sign perturbation of the uniform choice system, with the epsilon domain retained so the displayed masses are probabilities. Its incidence-level formula is the formal representation of source equation (4).
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion

Paper-local declaration:
SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion
Source locator: cited publication, lines 489-523

Verbatim paper-source connection

```
\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$, we
↪ begin by examining the cycle decompositions of $G_\mathcal{C}$.
    We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we
    ↪ call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to
    ↪ select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of
    ↪ $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

    \begin{figure}[t]
        \centering
        \includegraphics[height=55mm]{iia_graph_illustration.pdf}
        \caption{
            A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The
            ↪ hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph
            ↪ $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two
            ↪ undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles
            ↪ are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other
            ↪ orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
        }
        \label{fig:gc}
        \vspace{-4mm}
    \end{figure}
    A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$,
    ↪ and an element $\sigma_i \in \sigma$ denotes the $i$th cycle in $\sigma$. We use $|\sigma|$ to denote the number of cycles in the
    ↪ decomposition $\sigma$, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write
    ↪ $|\sigma_i|=2k_i$, where $k_i$ is the number of item-nodes in the cycle.

    Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation
    ↪ follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with $n$ item-nodes and $m$ set-nodes, any
    ↪ cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two
    ↪ smaller edge-disjoint cycles.

    \xhdr{From cycle decomposition to perturbations}
    We generate a structured collection of perturbation vectors $b$ by generating many distinct Eulerian orientations of a given cycle decomposition
    ↪ $\sigma$. We can independently direct the edges in each of the cycles in $\sigma$ in one of 2 directions (clockwise or counterclockwise),
    ↪ and all of these collections of directions correspond to valid $b$ vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation
    ↪ for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector $b$.

    We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors $b$ that orient a cycle decomposition $\sigma$, refining our generic
    ↪ definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
    Here $(x,C)$ indexes the elements of the vector $b$ and we use $\ell$ to index sequential edges in a cycle $\sigma_i$. For each perturbation vector
    ↪ $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) =
    ↪ \frac{d}{|\sigma|}$, the average cycle length.
```

Lean-expanded paper semantic target

```
type:
{ι : Type} → [Fintype ι] → ℕ → (ι → ℕ) → ℝ

value:
fun {ι : Type} [Fintype ι] (d : ℕ) (length : ι → ℕ) => 1 / ↑d * ∑ i : ι, ↑(length i) ^ 2
```

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

Reason: The target is the source cycle-dispersion index alpha(sigma)=(1/d) sum_i |sigma_i|^2, with d represented by the incidence count and the cycle lengths represented by the decomposition.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts

Paper-local declaration:
SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts
Source locator: cited publication, lines 419-456

Verbatim paper-source connection

For the first simple step, consider the testing problem:
\begin{align}
\label{eq:test2}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for } p = p_0 \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta,
\end{cases}
\end{align}
where p_0 is the uniform distribution, $p_{0,(x,C)} = 1/d, \forall (x,C)$ (recalling that each discete distribution is d-dimensional, where d is the sum of the cardinalities of the choice sets in a given $\mathcal C$). Since $p_0 \in \mathcal{P}_0$, the problem is simpler than the original problem, and any lower bound on the performance of this problem carries over to the original problem of interest.

Now consider a discrete distribution $\qbe$ that is a perturbation of the uniform distribution of the form:
\begin{align*}
\qbe((x,C)) = \frac{1}{d} + \frac{\epsilon b_{(x,C)}}{d},
\end{align*}
\begin{align*}
\epsilon \in [0,1] \
b \in \lbrace -1, 1 \rbrace^d,
\
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\text{ and }
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x.
\end{align*}

That is, b perturbs the uniform distribution such that there is no net translation over any set, or over all of the appearances of an item over sets. Our goal is to construct perturbations $\qbe$, selecting ϵ and b that land in $\Mdelta$, meaning they are at least δ away (in TV distance) from $\Piia$ but at the same time still correspond to valid probability distributions (i.e., within the d-simplex). We are starting our journey (in the direction of b) at the uniform distribution p_0, and we want to show that we always land at least δ away from {\it any} distribution in $\Piia$.

Let $\Bset$ be any collection of $M=|\Bset|$ vectors b satisfying the above conditions for the given collection $\mathcal C$. That is,
\begin{align}
\label{eq:bset_full}
\Bset \subseteq
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x
\right \}.
\end{align}
We define $\bar{q}_\epsilon = \frac{1}{M} \sum_{b \in \Bset} \qbe$ as the mixture distribution over the M distributions in $\Bset$. Samples from $\bar{q}_\epsilon$ would be drawn marginally, that is, some $\qbe$ would be chosen uniformly from the M possible distributions, and samples would then come from that $\qbe$. Consider then the testing problem:

Lean-expanded paper semantic target

```
type:
{α : Type} →
  [inst : Fintype α] →
    [DecidableEq α] →
      {β : Type} →
        (B : Finset β) →
          B.Nonempty →
            (β → SeshadriUgander2020IIATesting.FiniteDistribution α) →
              (N : ℕ) → SeshadriUgander2020IIATesting.FiniteDistribution (Fin N → α)

value:
fun {α : Type} [Fintype α] [DecidableEq α] {β : Type} (B : Finset β) (hB : B.Nonempty)
    (q : β → SeshadriUgander2020IIATesting.FiniteDistribution α) (N : ℕ) =>
  SeshadriUgander2020IIATesting.FiniteDistribution.uniformMixture B hB fun (b : β) => (q b).product N
```

Direct retained dependencies

- SeshadriUgander2020IIATesting.FiniteDistribution
- SeshadriUgander2020IIATesting.FiniteDistribution.product
- SeshadriUgander2020IIATesting.FiniteDistribution.uniformMixture

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

Reason: The source lower-bound experiment averages the N-fold product distributions generated by balanced sign perturbations. The target formalizes exactly this finite mixture of product measures, with the mixture index exposing the source random-sign construction.
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 595-604

Verbatim source input

```
\begin{lemma}
        \label{lemma:bound1}
        $$
        \chi^2(\mathbb{P}_1,\mathbb{P}_0) + 1
        \le
        \frac{1}{M^2}\sum_{b,b' \in \Bset}\exp\Big(\frac{N \epsilon^2}{d}b^Tb'\Big)
        ,
        $$
        where $\Bset$ is a set of arbitrary perturbations $b$ of size $|\Bset|=M$ satisfying $b \in \lbrace -1, 1 \rbrace^d$, $\sum_{\substack{C \in
        ↪  \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x$, and $\sum_{y \in C} b_{y,C} = 0, \forall C$.
    \end{lemma}

[Next verbatim source excerpt]

    For the first simple step, consider the testing problem:
    \begin{align}
    \label{eq:test2}
    \begin{cases}
    H_0: (x, C) \sim p^N & \text{for } p = p_0 \\
    H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta,
    \end{cases}
    \end{align}
    where $p_0$ is the uniform distribution, $p_{0,(x,C)} = 1/d, \forall (x,C)$ (recalling that each discete distribution is $d$-dimensional, where $d$ is the
    ↪  sum of the cardinalities of the choice sets in a given $\mathcal C$). Since $p_0 \in \mathcal{P}_0$, the problem is simpler than the original
    ↪  problem, and any lower bound on the performance of this problem carries over to the original problem of interest.

    Now consider a discrete distribution $\qbe$ that is a perturbation of the uniform distribution of the form:
    \begin{align*}
    \qbe((x,C)) = \frac{1}{d} + \frac{\epsilon b_{(x,C)}}{d},
    \end{align*}
    \begin{align*}
    \epsilon \in [0,1] \
    b \in \lbrace -1, 1 \rbrace^d,
    \
    \sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
    \text{ and }
    \sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x.
    \end{align*}

    That is, $b$ perturbs the uniform distribution such that there is no net translation over any set, or over all of the appearances of an item over sets.
    ↪  Our goal is to construct perturbations $\qbe$, selecting $\epsilon$ and $b$ that land in $\Mdelta$, meaning they are at least $\delta$ away (in
    ↪  TV distance) from $\Pia$ but at the same time still correspond to valid probability distributions (i.e., within the $d$-simplex). We are starting our
    ↪  journey (in the direction of $b$) at the uniform distribution $p_0$, and we want to show that we always land at least $\delta$ away from {\it
    ↪  any} distribution in $\Pia$.

    Let $\Bset$ be any collection of $M=|\Bset|$ vectors $b$ satisfying the above conditions for the given collection $\mathcal C$. That is,
    \begin{align}
    \label{eq:bset_full}
    \Bset \subseteq
    \left \{
    b \in \lbrace -1, 1 \rbrace^d
    :
    \sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
    \ \
    \sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x
    \right \}.
    \end{align}
    We define $\bar{q}_\epsilon = \frac{1}{M} \sum_{b \in \Bset} \qbe$ as the mixture distribution over the $M$ distributions in $\Bset$. Samples from
    ↪  $\bar{q}_\epsilon$ would be drawn marginally, that is, some $\qbe$ would be chosen uniformly from the $M$ possible distributions, and
    ↪  samples would then come from that $\qbe$. Consider then the testing problem:
```

Expanded PaperInterface specification

```
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} {β : Type} (B : Finset β) (hB : B.Nonempty)
  (b : β → SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F) (ε : ℝ) (hε_nonneg : 0 ≤ ε) (hε_le_one : ε ≤ 1)
  (N : ℕ),
  (SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts B hB
          (fun (z : β) =>
            (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (b z) ε hε_nonneg hε_le_one).asFiniteDistribution)
          N).chiSquare
      ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N) +
    1 ≤
  1 / ↑B.card ^ 2 *
    ∑ z ∈ B,
      ∑ z' ∈ B,
        Real.exp
          (↑N * (ε ^ 2 / ↑F.incidenceCount * SeshadriUgander2020IIATesting.ChoiceSystem.signInner (b z) (b z')))
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame

- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceFrame.instDecidableEqObservation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign
- SeshadriUgander2020IIATesting.ChoiceSystem.asFiniteDistribution
- SeshadriUgander2020IIATesting.ChoiceSystem.perturb
- SeshadriUgander2020IIATesting.ChoiceSystem.signInner
- SeshadriUgander2020IIATesting.ChoiceSystem.uniform
- SeshadriUgander2020IIATesting.FiniteDistribution.chiSquare
- SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts
- SeshadriUgander2020IIATesting.FiniteDistribution.product

Lean-elaborated claim roles

```
1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. parameter: Type
3. parameter: Finset β
4. assumption: B.Nonempty
5. parameter: β → SeshadriUgander2020IIATesting.ChoiceSystem.BalancedSign F
6. parameter: ℝ
7. assumption: 0 ≤ ε
8. assumption: ε ≤ 1
9. parameter: ℕ
10. conclusion: (SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts B hB
        (fun (z : β) =>
          (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (b z) ε hε_nonneg hε_le_one).asFiniteDistribution)
        N).chiSquare
    ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N) +
  1 ≤
 1 / ↑B.card ^ 2 *
  ∑ z ∈ B,
    ∑ z' ∈ B,
      Real.exp (↑N * (ε ^ 2 / ↑F.incidenceCount * SeshadriUgander2020IIATesting.ChoiceSystem.signInner (b z) (b z')))
```

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.lemma3_chiSquare_mixture

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

Reason: The target quantifies over an arbitrary nonempty finite family B of balanced sign perturbations, identifies M with card(B), and gives the same chi-square-plus-one upper bound by the double sum of exp((N epsilon^2/d) b^T b'). The explicit epsilon-in-[0,1] hypotheses are the source perturbation's probability-domain conditions.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 2

Source locator: cited publication, lines 631-638

Verbatim source input

```
\begin{lemma}
	\label{lemma:bound2}
	\begin{align*}
	\chi^2(\mathbb{P}_1,\mathbb{P}_0) + 1
	\ \ \le \exp\Big(\frac{N^2 \epsilon^4}{2d} \alpha(\sigma) \Big),
	\end{align*}
	for any cycle decomposition $\sigma \in \Sigma_\mathcal{C}$.
\end{lemma}
```

[Next verbatim source excerpt]

```
\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$,
↪ we begin by examining the cycle decompositions of $G_\mathcal{C}$.
We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we
↪ call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to
↪ select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of
↪ $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

\begin{figure}[t]
	\centering
	\includegraphics[height=55mm]{iia_graph_illustration.pdf}
	\caption{
		A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The
		↪ hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph
		↪ $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two
		↪ undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles
		↪ are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other
		↪ orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
	}
	\label{fig:gc}
	\vspace{-4mm}
\end{figure}
A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$,
↪ and an element $\sigma_i \in \sigma$ denotes the $i$th cycle in $\sigma$. We use $|\sigma|$ to denote the number of cycles in the
↪ decomposition $\sigma$, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write
↪ $|\sigma_i|=2k_i$, where $k_i$ is the number of item-nodes in the cycle.

Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation
↪ follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with $n$ item-nodes and $m$ set-nodes, any
↪ cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two
↪ smaller edge-disjoint cycles.

\xhdr{From cycle decomposition to perturbations}
We generate a structured collection of perturbation vectors $b$ by generating many distinct Eulerian orientations of a given cycle decomposition
↪ $\sigma$. We can independently direct the edges in each of the  cycles in $\sigma$ in one of 2 directions (clockwise or counterclockwise),
↪ and all of these collections of directions correspond to valid $b$ vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation
↪ for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector $b$.

We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors $b$ that orient a cycle decomposition $\sigma$, refining our generic
↪ definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
Here $(x,C)$ indexes the elements of the vector $b$ and we use $\ell$ to index sequential edges in a cycle $\sigma_i$. For each perturbation vector
↪ $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) =
↪ \frac{d}{|\sigma|}$, the average cycle length.
```

[Next verbatim source excerpt]

```
The remaining analysis applies only to sets of perturbations $\Bset(\sigma)$ derived from an Eulerian orientation $\sigma$ of $G_\mathcal{C}$. With
↪ such a collection of perturbations, we can bound the role of $b^Tb'$ for each pair of perturbations. Recall that $\Bset(\sigma)$ was defined in
↪ \eqref{eq:bset} as the collection of $2^{|\sigma|}$ Eulerian orientations associated with a particular cycle decomposition $\sigma$. We now
↪ additionally define a set of vectors $\mathcal{A}_\mathcal{C}(\sigma)$, such that for every element $b \in \mathcal{B}_\mathcal{C}(\sigma)$,
↪ $\mathcal{A}_\mathcal{C}(\sigma)$ contains a vector $a \in \lbrace -1, 1\rbrace^{|\sigma|}$ to denote the directions of the $|\sigma|$ cycles
↪ that created $b$. Moreover, we introduce for every $\sigma$ a quantity $\alpha(\sigma) = \frac{1}{d} \sum_{\sigma_i \in \sigma}|\sigma_i|^2$
↪ that serves as a normalized measure of the ``dispersion'' of the cycle decomposition $\sigma$. With this notation, we proceed to the following
↪ result.
\begin{lemma}
	\label{lemma:bound2}
	\begin{align*}
	\chi^2(\mathbb{P}_1,\mathbb{P}_0) + 1
	\ \ \le \exp\Big(\frac{N^2 \epsilon^4}{2d} \alpha(\sigma) \Big),
	\end{align*}
	for any cycle decomposition $\sigma \in \Sigma_\mathcal{C}$.
\end{lemma}
\begin{proof}
```

We first seek to control the value b^Tb'. Using the vectors $a \in \mathcal{A}_\mathcal{C}(\sigma)$, we have that $b^Tb' = \sum_{i \in s} |\sigma_i| a_i a'_i$, where a corresponds to the direction of the cycles for the vector b, and a' for b'.

Recalling that we chose our $M = 2^{|\sigma|}$ perturbations based on orienting the cycle decomposition σ, we begin where Lemma~\ref{lemma:bound1} left off and have

$$
\begin{aligned}
&\chi^2(\mathbb{P}_1,\mathbb{P}_0) + 1 \\
&\leq \frac{1}{M^2}\sum_{b,b'}\exp\Big(\frac{N \epsilon^2}{d}b^Tb'\Big)\\
&=\frac{1}{2^{2|\sigma|}}\sum_{a,a'}\exp\Big(\frac{N \epsilon^2}{d}\sum_{\sigma_i \in \sigma} |\sigma_i| a_i a'_i \Big)\\
&=\mathbb{E}\Big[\prod_{\sigma_i \in \sigma}\exp\Big(\frac{N \epsilon^2}{d} |\sigma_i| a_i a'_i \Big)\Big]\\
&=\prod_{\sigma_i \in \sigma}\mathbb{E}\Big[\exp\Big(\frac{N \epsilon^2}{d} |\sigma_i| a_i a'_i \Big)\Big]\\
&=\prod_{\sigma_i \in \sigma}\Big(\frac{1}{2}\exp\Big(\frac{N \epsilon^2}{d}|\sigma_i| \Big) + \frac{1}{2}\exp\Big(\frac{-N \epsilon^2}{d}|\sigma_i| \Big)\Big)\\
&\leq \prod_{\sigma_i \in \sigma}\Big(\exp\Big(\frac{N^2 \epsilon^4}{2d^2}(|\sigma_i|)^2 \Big)\Big)\\
&=\exp\Big(\frac{N^2 \epsilon^4}{2d}\alpha(\sigma) \Big),
\end{aligned}
$$

where the second line follows from controlling b^Tb' with a vectors, the third line follows from the second line being equivalent to the expectation over every pair of vectors a, a', the fourth line follows from the fact that each element of vector a is independent from the other elements, the fifth line follows from evaluating the expectation for a pair of independent Rademacher variables, the sixth line follows from the fact that $\frac{1}{2}\exp(x) + \frac{1}{2}\exp(-x) \leq \exp(\frac{x^2}{2})$, and the final line follows from rearranging the terms and applying the definition of $\alpha(\sigma)$.

Expanded PaperInterface specification

```
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F)
  (ε : ℝ) (hε_nonneg : 0 ≤ ε) (hε_le_one : ε ≤ 1) (N : ℕ),
  (SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts Finset.univ
         (SeshadriUgander2020IIATesting.lemma2_testing_reductionSpec._proof_2 D)
         (fun (a : D.Cycle → Bool) =>
           (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign a) ε hε_nonneg
              hε_le_one).asFiniteDistribution)
         N).chiSquare
     ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N) +
    1 ≤
   Real.exp
    (↑N ^ 2 * ε ^ 4 / (2 * ↑F.incidenceCount) *
     SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length)
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceFrame.instDecidableEqObservation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.orientedSign
- SeshadriUgander2020IIATesting.ChoiceSystem.asFiniteDistribution
- SeshadriUgander2020IIATesting.ChoiceSystem.perturb
- SeshadriUgander2020IIATesting.ChoiceSystem.uniform
- SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion
- SeshadriUgander2020IIATesting.FiniteDistribution.chiSquare
- SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts
- SeshadriUgander2020IIATesting.FiniteDistribution.product

Lean-elaborated claim roles

1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. parameter: SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F
3. parameter: ℝ
4. assumption: 0 ≤ ε
5. assumption: ε ≤ 1
6. parameter: ℕ
7. conclusion: (SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts Finset.univ
   (SeshadriUgander2020IIATesting.lemma2_testing_reductionSpec._proof_2 D)
   (fun (a : D.Cycle → Bool) =>
     (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign a) ε hε_nonneg
        hε_le_one).asFiniteDistribution)
   N).chiSquare
  ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N) +
 1 ≤
Real.exp
 (↑N ^ 2 * ε ^ 4 / (2 * ↑F.incidenceCount) *
  SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length)

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.lemma4_cycle_chiSquare

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

Reason: For every cycle decomposition, perturbation level in the source domain, and sample size, the target bounds chi-square plus one by exp(N^2 epsilon^4 alpha(sigma)/(2d)). Its incidenceCount is d and cycleDispersion is the source alpha(sigma), so the mathematical statement and constant match.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 3

Source locator: cited publication, lines 321-335

Verbatim source input

```
\begin{theorem}\label{thm:lower_bound}
    Up to some constant $c_1$ and properties $\mu(\sigma)$ and $\alpha(\sigma)$ of any cycle decomposition $\sigma$ of the comparison
    ↪ incidence graph $G_\mathcal{C}$, the minimax risk $R_{N,\delta}(\Piia)$ is lower bounded as
    $$
    \frac{1}{2} -
    \frac{1}{4}
    \Big(\exp\Big(
    \frac{8\mu(\sigma)^4\alpha(\sigma)N^2 \delta^4}{d}\Big) -1
    \Big)^{\frac{1}{2}}
    \leq R_{N,\delta}(\Piia).
    $$
    The testing radius $\delta_N(\Piia)$ and sample complexity $N_\delta(\Piia)$ then scale as at least
    $$
    \delta_N(\Piia) \GtrSim \frac{d^{\frac{1}{4}}}{\mu(\sigma)\alpha(\sigma)^{\frac{1}{4}}\sqrt{N}}, \ \ \ \
    N_\delta(\Piia) \GtrSim \frac{\sqrt{d}}{\sqrt{\mu(\sigma)^4\alpha(\sigma)}\delta^2}.$$
    \end{theorem}
```

[Next verbatim source excerpt]

```
    We call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by
    ↪ $\sigma$. We define two functions of $\sigma$ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle
    ↪ length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion
    ↪ index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to
    ↪ a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms
    ↪ further in Section \ref{sec:bounding}.

    For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$
    ↪ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in
    ↪ $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a
    ↪ consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
    We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do
    ↪ not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes
    ↪ the requirement of even set sizes and even item appearances.

    \section{Lower bound on the minimax risk}
    \label{sec:lowerbounds}

    Given a dataset $\mathcal D_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ of choices, consider the testing problem, as stated in \eqref{eq:iiatest}, of the
    ↪ IIA property against the alternative of an arbitrary choice system that is $\delta$-separated from IIA. We seek to lower bound the minimax risk of
    ↪ this testing problem. The maximum risk of any test $\phi$ is a useful quantity that demonstrates how poor the test could be at finding IIA
    ↪ violations without further prior knowledge about the structure of the violations. Because tests can provide high power over certain types of
    ↪ violations while providing no power over others, it provides a level standard of comparison across tests by studying each test's worst case
    ↪ scenario \cite{balakrishnan2018hypothesis}.

    A minimax lower bound then bounds the error of the best possible test in its worst case, from below. The bound is given as a function of the number
    ↪ of samples $N$, the separation $\delta$ between the null and the alternative, and problem parameters. It is a statement about the fundamental
    ↪ possibility (or limitation) of effective tests for the testing problem. A lower bound on the minimax risk can also be restated as a lower bound on
    ↪ the sample complexity---the number of samples required to achieve a probability of error for a given separation---and equivalently, as a lower
    ↪ bound on the testing radius---the separation required to perform a test with a certain probability of error given a specific number of samples.

    \subsection{Minimax risk preliminaries}
    The minimax risk $R_{N,\delta}(\Piia)$ of the IIA testing problem, where risk is used in a uniform sense between the null and the alternative, is then
    \begin{align*}
    R_{N,\delta}(\Piia) = \inf_{\phi} \sup_{p \in \Piia, q \in \Mdelta} \frac{1}{2}p^N(\phi(\mathcal{D}_N) = 1) + \frac{1}{2}q^N(\phi(\mathcal{D}_N) =
    ↪ 0),
    \end{align*}
    where the $\inf$ is taken over any test $\phi$. The testing radius $\delta_N(\Piia)$ is then be defined as
    \begin{align*}
    \delta_N(\Piia) \defeq \inf \lbrace \delta \mid R_{N,\delta}(\Piia) \leq a \rbrace,
    \end{align*}
    where $a$ is any constant less than $1/2$ (sometimes arbitrarily set to $a=1/3$). The testing radius describes the smallest size of indifference zone
    ↪ that allows hypotheses to be uniformly distinguishable with probability $1-a$.
    Finally, the sample complexity $N_\delta(\Piia)$ is defined similarly as
    \begin{align*}
    N_\delta(\Piia) \defeq \inf \lbrace N \mid R_{N,\delta}(\Piia) \leq a \rbrace.
    \end{align*}

    For both the sample complexity and testing radius quantities, we at times use $\GtrSim$, followed by a expression, to denote that the quantities
    ↪ scale at at least a constant times that expression. As an alternative objective to lower bounding the minimax risk $R_{N,\delta}(\Piia)$, we
    ↪ provide in the appendix (Section~\ref{app:alphatest}) a reformulation of risk bounding as a problem centered around level-$\alpha$ tests, with
    ↪ the corresponding parallel statement to the risk-centered Theorem~\ref{thm:lower_bound} given below.
```

[Next verbatim source excerpt]

```
    For the first simple step, consider the testing problem:
    \begin{align}
    \label{eq:test2}
    \begin{cases}
    H_0: (x, C) \sim p^N & \text{for } p = p_0 \\
    H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta,
    \end{cases}
    \end{align}
    where $p_0$ is the uniform distribution, $p_{0,(x,C)} = 1/d, \forall (x,C)$ (recalling that each discete distribution is $d$-dimensional, where $d$ is the
    ↪ sum of the cardinalities of the choice sets in a given $\mathcal C$). Since $p_0 \in \mathcal{P}_0$, the problem is simpler than the original
    ↪ problem, and any lower bound on the performance of this problem carries over to the original problem of interest.

    Now consider a discrete distribution $\qbe$ that is a perturbation of the uniform distribution of the form:
```

\begin{align*}
\qbe((x,C)) = \frac{1}{d} + \frac{\epsilon b_{(x,C)}}{d},
\end{align*}
\begin{align*}
\epsilon \in [0,1] \
b \in \lbrace -1, 1 \rbrace^d,
\
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\text{ and }
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x.
\end{align*}

That is, b perturbs the uniform distribution such that there is no net translation over any set, or over all of the appearances of an item over sets.
↪ Our goal is to construct perturbations $\qbe$, selecting ϵ and b that land in $\Mdelta$, meaning they are at least δ away (in
↪ TV distance) from $\Piia$ but at the same time still correspond to valid probability distributions (i.e., within the d-simplex). We are starting our
↪ journey (in the direction of b) at the uniform distribution p_0, and we want to show that we always land at least δ away from {\it
↪ any} distribution in $\Piia$.

Let $\Bset$ be any collection of $M=|\Bset|$ vectors b satisfying the above conditions for the given collection $\mathcal C$. That is,
\begin{align}
\label{eq:bset_full}
\Bset \subseteq
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x
\right \}.
\end{align}
We define $\bar{q}_\epsilon = \frac{1}{M} \sum_{b \in \Bset} \qbe$ as the mixture distribution over the M distributions in $\Bset$. Samples from
↪ $\bar{q}_\epsilon$ would be drawn marginally, that is, some $\qbe$ would be chosen uniformly from the M possible distributions, and
↪ samples would then come from that $\qbe$. Consider then the testing problem:

[Next verbatim source excerpt]

\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$,
↪ we begin by examining the cycle decompositions of $G_\mathcal{C}$.
We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we
↪ call a decomposition of $G_\mathcal{C}$ into a set of s simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to
↪ select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of
↪ $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

\begin{figure}[t]
\centering
\includegraphics[height=55mm]{iia_graph_illustration.pdf}
\caption{
A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The
↪ hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph
↪ $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two
↪ undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles
↪ are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other
↪ orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
}
\label{fig:gc}
\vspace{-4mm}
\end{figure}
A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$,
↪ and an element $\sigma_i \in \sigma$ denotes the ith cycle in σ. We use $|\sigma|$ to denote the number of cycles in the
↪ decomposition σ, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write
↪ $|\sigma_i|=2k_i$, where k_i is the number of item-nodes in the cycle.

Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation
↪ follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with n item-nodes and m set-nodes, any
↪ cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two
↪ smaller edge-disjoint cycles.

\xhdr{From cycle decomposition to perturbations}
We generate a structured collection of perturbation vectors b by generating many distinct Eulerian orientations of a given cycle decomposition
↪ σ. We can independently direct the edges in each of the cycles in σ in one of 2 directions (clockwise or counterclockwise),
↪ and all of these collections of directions correspond to valid b vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation
↪ for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector b.

We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors b that orient a cycle decomposition σ, refining our generic
↪ definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
Here (x,C) indexes the elements of the vector b and we use ℓ to index sequential edges in a cycle σ_i. For each perturbation vector
↪ $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) =
↪ \frac{d}{|\sigma|}$, the average cycle length.

[Next verbatim source excerpt]

\section{Proof of main result}

```
\label{sec:thmproof}

We now use Lemma~\ref{lemma:bound1} and \ref{lemma:bound2} to prove the main result in Theorem~\ref{thm:lower_bound}.

\begin{proof}
    \begin{align*}
    R_{N,\delta}(\Piia)
    &\geq \frac{1}{2} - \frac{1}{4}||\mathbb{P}_0 - \mathbb{P}_1||_{TV}
    \geq
    \frac{1}{2} - \frac{1}{4}\sqrt{\chi^2(\mathbb{P}_1,\mathbb{P}_0)} \\
    &\geq \frac{1}{2} - \frac{1}{4}\Big(\exp\Big(\frac{N^2 \epsilon^4}{2d}\alpha(\sigma) \Big) - 1\Big)^{\frac{1}{2}}  \\
    &\geq
    \frac{1}{2} - \frac{1}{4}\Big(\exp\Big(\frac{8\mu(\sigma)^4\alpha(\sigma)N^2 \delta^4}{d} \Big) - 1\Big)^{\frac{1}{2}},
    \end{align*}
    where the last step follows from substituting $\epsilon = 2\mu(\sigma)\delta$. The final expression makes clear the dependence of the lower
    ↪  bound on $\sigma$. Namely, the bound  applies to any $\sigma$ of  $G_\mathcal{C}$, and so $\sigma$ can be chosen to produce the best
    ↪  lower bound. This observation means we can restate the risk bound as
    \begin{align*}
    R_{N,\delta}(\Piia)
    &\geq \max_{\sigma \in \Sigma_{\mathcal{C}}} \frac{1}{2} - \frac{1}{4}\Big(\exp\Big(\frac{8\mu(\sigma)^4\alpha(\sigma)N^2 \delta^4}{d}
    ↪  \Big) - 1\Big)^{\frac{1}{2}},
    \end{align*}
    where we remind the reader that $\Sigma_\mathcal{C}$ is the collection of all possible edge-disjoint cycle decompositions of $G_\mathcal{C}$.
    Indeed, the smaller the size of each cycle in the decomposition of $G_\mathcal{C}$, the stronger the lower bound gets, since both
    ↪  $\mu(\sigma)$, the average cycle size, and $\alpha(\sigma)$, the cycle dispersion index, become smaller.
\end{proof}
```

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

```
For every concrete comparison-incidence cycle decomposition, the finite minimax-risk lower bound holds under the perturbation-domain condition 2
↪  mu(sigma) delta <= 1, and any uniformly quarter-accurate test obeys the displayed exact testing-radius and sample-complexity lower bounds.
```

Archival source anchor: cited publication, lines 321-335

Expanded PaperInterface specification

```
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F)
  (_W : D.AlternatingCycleWitness) (δ : ℝ),
  0 < δ →
    2 * D.cycleMean * δ ≤ 1 →
      ∀ (N : ℕ),
        0 < N →
          SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound N δ
            (1 / 2 -
              1 / 4 *
                √(Real.exp
                    (8 * D.cycleMean ^ 4 *
                          SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length *
                        ↑N ^ 2 *
                      δ ^ 4 /
                    ↑F.incidenceCount) -
                  1)) ∧
          (SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest N δ →
            √(↑F.incidenceCount * Real.log 2 /
                (8 * D.cycleMean ^ 4 *
                   SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length *
                  δ ^ 4)) ≤
              ↑N) ∧
            (SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest N δ →
              √√(↑F.incidenceCount * Real.log 2 /
                    (8 * D.cycleMean ^ 4 *
                       SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length *
                      ↑N ^ 2)) ≤
                δ)
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.AlternatingCycleWitness
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean
- SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest
- SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound
- SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion

Lean-elaborated claim roles

```
1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. parameter: SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F
3. parameter: D.AlternatingCycleWitness
4. parameter: ℝ
5. assumption: 0 < δ
6. assumption: 2 * D.cycleMean * δ ≤ 1
7. parameter: ℕ
8. assumption: 0 < N
9. conclusion: SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound N δ
   (1 / 2 -
     1 / 4 *
       √(Real.exp
           (8 * D.cycleMean ^ 4 *
                SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length *
              ↑N ^ 2 *
            δ ^ 4 /
          ↑F.incidenceCount) -
         1)) ∧
 (SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest N δ →
    √(↑F.incidenceCount * Real.log 2 /
        (8 * D.cycleMean ^ 4 *
           SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length *
         δ ^ 4)) ≤
      ↑N) ∧
   (SeshadriUgander2020IIATesting.ChoiceSystem.HasQuarterAccurateTest N δ →
     √√(↑F.incidenceCount * Real.log 2 /
           (8 * D.cycleMean ^ 4 *
              SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length *
            ↑N ^ 2)) ≤
       δ)
```

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.theorem1_testing_lower_bound

Recorded source-to-Spec screening

Verdict: matches_approved_corrected_target
Reason: The target retains the source finite minimax-risk lower bound with the same exponent 8 mu(sigma)^4 alpha(sigma) N^2 delta^4/d and its quarter-error form. The explicit alternating witness and 2 mu(sigma) delta <= 1 premise are the approved clarification of the source perturbation’s probability-domain condition.
Reviewer check: ☐ Matches formalized target
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 4

Source locator: cited publication, lines 680-687

Verbatim source input

```
\begin{cor}\label{cor:global_lower_bound}
        For any $G_\mathcal{C}$, $R_{N,\delta}(\Piia)$ may be globally lower bounded as
        \begin{align*}
        R_{N,\delta}(\Piia) \geq \frac{1}{2} - \frac{1}{4}\Big(\exp\Big(
        c \min \Big\{\frac{(d^4\log(n)^5 + d^3(n-\log(n))n\log(n)^4)N^2 \delta^4}{d(d - 4n + 8\log(n))^4}, \frac{n^5N^2 \delta^4}{d} \Big\}
        \Big) - 1\Big)^{\frac{1}{2}}.
        \end{align*}
    \end{cor}
```

[Next verbatim source excerpt]

```
        \begin{lemma}\label{lemma:Gc_cycle_decomposition}
            Any Eulerian comparison incidence graph $G_\mathcal{C}$, defined in Section \ref{subsection:comparison_graph}, has a cycle
            ↪ decomposition $\sigma$ where all but at most $\min(2n+m, 4n)$ edges (i.e., at most $\max(d-\min(2n+m, 4n), 0)$ edges) contribute
            ↪ to cycles of size at most $2\floor{2 \log_2(n)}$ and the rest (at most $\min(2n+m, 4n)$) contribute to cycles of size at most $2n$. We
            ↪ then have the following bounds on $\mu(\sigma)$ and $\alpha(\sigma)$:
            \begin{align*}
            \mu(\sigma) \leq \min \Big\{\Big(\frac{d}{d - 4n + 8\log_2 (n)}\Big)4\log_2 (n), 2n\Big\} \\
            \alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d}, 2n\Big\}.
            \end{align*}
        \end{lemma}
        \begin{proof}
            Apply Lemma \ref{lemma:bipartite_cycle_decomposition} to $G_\mathcal{C}$, with $n_1 = n$ and $n_2 = m$. Since $G_\mathcal{C}$ is
            ↪ Eulerian, $n_2^\text{(odd)} = 0$, and so we know that $G_C$ has a cycle decomposition with cycles of size at most $2 \floor{2
            ↪ \log(n_1)}$ with at most $\min(2n + m, 4n)$ edges that remain. Again since $G_\mathcal{C}$ is Eulerian, and removing cycles from
            ↪ $G_\mathcal{C}$ does not change this fact, the remaining $\min(2n + m, 4n)$ edges can always be decomposed into cycles of size at
            ↪ most $2n$. The latter bound on the cycle size follows from the application of the pigeonhole principle described in Section
            ↪ $\ref{sec:relax}$. When combined, these steps form the $sigma$ that possesses the properties stated in the Lemma.

            Naturally, the short cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are only guaranteed if $d$ exceeds $\min(2n + m,
            ↪ 4n)$, else we use the extremal bound on cycle size of $2n$ for all edges in $d$. Note that $2\floor{2 \log_2(n)} \leq 2n$ for all $n \in
            ↪ \mathbb{N}$, so the cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are always smaller than those guaranteed by the
            ↪ extremal bound. We then have the following lower bound on $|\sigma|$, the number of cycles in the cycle decomposition:
            \begin{align*}
            |\sigma| \geq \max \Big\{\frac{d - 2n - \min(2n, m)}{2\floor{2 \log_2(n)}} + \frac{2n + \min(2n, m)}{2n}, \frac{d}{2n}\Big\}.
            \end{align*}
            We replace $\min(2n,m)$ with $2n$ to weaken the above bound at the benefit of making the expression much simpler, especially for its
            ↪ use bounding $\mu(\sigma)$ and $\alpha(\sigma)$. We also weaken $2\floor{2 \log_2(n)}$ to $4 \log_2(n)$ for the same purpose. For
            ↪ the sharpest results, we invite the reader to carry the original bound over to $\mu(\sigma)$ and $\alpha(\sigma)$ in a manner similar
            ↪ to what we do below. We then have:
            \begin{align*}
            |\sigma| \geq \max \Big\{\frac{d - 4n}{4\log_2 (n)} + 2, \frac{d}{2n}\Big\}.
            \end{align*}
            Since $\mu(\sigma) = \frac{d}{|\sigma|}$, the preceding bound immediately yields an upper bound on $\mu(\sigma)$:
            \begin{align*}
            \mu(\sigma) \leq \frac{d}{\max \Big\{\frac{d - 4n}{4\log_2(n)} + 4, \frac{d}{2n}\Big\}} = \min \Big\{\Big(\frac{d}{d - 4n + 8\log_2
            ↪ (n)}\Big)4\log_2 (n), 2n\Big\}.
            \end{align*}
            Now, we bound $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$. Clearly, $\alpha(\sigma)$ grows with the size of
            ↪ the cycles in $\sigma$, so we may upper bound it by considering the largest size the cycles in $\sigma$ could be. Recall that
            ↪ $2\floor{2 \log_2(n)} \leq 2n$ for all $n \in \mathbb{N}$, so the cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are
            ↪ always smaller than those guaranteed by the extremal bound of $2n$. Thus, the more size-$2n$ cycles in our cycle decomposition, the
            ↪ higher $\alpha(\sigma)$. Since we can have at most $4n$ edges contribute to cycles of size at most $2n$, if $d \leq 4n$, then we have:
            \begin{align*}
            \alpha(\sigma) \leq \frac{1}{d} \frac{d}{2n} (2n)^2 = 2n.
            \end{align*}
            Otherwise, we have:
            \begin{align*}
            \alpha(\sigma) \leq \frac{1}{d} \Big(\frac{d - 4n}{2\floor{2 \log_2(n)}}(2\floor{2 \log_2(n)})^2 + \frac{4n}{2n}(2n)^2 \Big) \leq 4
            ↪ \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d},
            \end{align*}
            where the first inequality follows from computing the formula with all $4n$ edges in cycles at their maximum size of $2n$ and the
            ↪ remaining edges in cycles at their maximum size of $2\floor{2 \log_2(n)}$. The second inequality comes from rearranging terms and
            ↪ weakening $2\floor{2 \log_2(n)}$ to $4 \log_2(n)$. Putting the two cases together, we have:
            \begin{align*}
            \alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d}, 2n\Big\}.
            \end{align*}
        \end{proof}
```

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

For an Eulerian comparison-incidence frame with at least four items and positive rational-branch denominator, the source global risk lower bound holds
↪ under the explicit perturbation-domain condition.

Archival source anchor: cited publication, lines 680-687

Expanded PaperInterface specification

```
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame},
  F.Eulerian →
    4 ≤ Fintype.card F.Item →
      0 < ↑F.incidenceCount - 4 * ↑(Fintype.card F.Item) + 2 * (4 * Real.logb 2 ↑(Fintype.card F.Item)) →
        ∀ (δ : ℝ),
```

```
0 ≤ δ →
 2 * SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound * δ ≤ 1 →
  ∀ (N : ℕ),
   SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound N δ
    (1 / 2 -
      1 / 4 *
       √(Real.exp
           (8 * SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound ^ 4 *
                 SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound *
               ↑N ^ 2 *
             δ ^ 4 /
           ↑F.incidenceCount) -
         1))
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Eulerian
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound
- SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound
- SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound

Lean-elaborated claim roles

1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. assumption: F.Eulerian
3. assumption: 4 ≤ Fintype.card F.Item
4. assumption: 0 < ↑F.incidenceCount - 4 * ↑(Fintype.card F.Item) + 2 * (4 * Real.logb 2 ↑(Fintype.card F.Item))
5. parameter: ℝ
6. assumption: 0 ≤ δ
7. assumption: 2 * SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound * δ ≤ 1
8. parameter: ℕ
9. conclusion: SeshadriUgander2020IIATesting.ChoiceSystem.ProductTestingLowerBound N δ

```
(1 / 2 -
  1 / 4 *
   √(Real.exp
       (8 * SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound ^ 4 *
             SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound *
           ↑N ^ 2 *
         δ ^ 4 /
       ↑F.incidenceCount) -
     1))
```

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.corollary1_global_lower_bound_corrected

Recorded source-to-Spec screening

Verdict: matches_approved_corrected_target

Reason: The expanded target states the source global testing-risk lower bound in its cycle-mean/cycle-dispersion form and retains the displayed risk expression. It makes the approved positive-denominator and perturbation-domain conditions explicit, so it matches the documented corrected target rather than the unrestricted printed presentation.

Reviewer check: ☐ Matches formalized target

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 5

Source locator: cited publication, lines 783-785

Verbatim source input

```
\begin{fact}\label{cvx_hull_dense}
            The convex hull of $\Piia$ is dense in $\Pall$.
        \end{fact}
```

[Next verbatim source excerpt]

```
\subsection{Choice systems}\label{subsection:choice_system}
Let $\mathcal{X}$ be a finite set of $n$ \textit{items} with generic element $x \in \mathcal{X}$, and let $\mathcal{C} \subseteq 2^\mathcal{X}$
↪ denote the observed space, a collection of $m \geq n$ unique unordered subsets $C \subseteq \mathcal{X}$ of size 2 or greater. A decision
↪ maker who is presented with a \textit{choice set} $C \in \mathcal{C}$ chooses one item from the set. We let $P_{x,C}$ denote the probability
↪ that $x$ is chosen from $C$, $\forall x \in C$, and let $w(C)$ denote the probability of seeing choice set $C$, $\forall C \in \mathcal{C}$. The
↪ object $\mathcal{C}$ should be treated as a sample frame---the regime of sets of interest, or the regime of sets for which a notion of error is
↪ important---determined ahead of time (as opposed to being defined in terms of the sets so far observed in some dataset). That is, we are
↪ operating in a fixed-design setting, where $\mathcal{C}$ does not depend on, e.g., prior observations.

Our test is focused on a given dataset $\mathcal{D}_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ that results from a decision maker making choices: a
↪ datapoint $j$ represents a decision scenario, and contains $C_j$, the choice set provided in that decision, and $x_j \in C_j$, the item chosen from
↪ that choice set. We assume each datapoint is an independent sample from the joint distribution over choices sets and choices:
\begin{align*}
Pr[(x_j, C_j) = (x, C)] = w(C) \cdot P_{x,C}, \ \ \forall j.
\end{align*}
That is, a datapoint is a sample from some discrete distribution over $d$ possible (choice, choice set) pairs, where $d = \sum_{C \in \mathcal{C}}
↪ |C|$. We refer to this $d$-dimensional discrete distribution as a \textit{choice system} and denote it using $q$.
Let $\Pall$ denote the collection of all choice systems on the collection $\mathcal{C}$, i.e., the collection of all $d$-dimensional discrete
↪ distributions. The (choice, choice set) tuples $(x,C)$ can be used to intuitively index this distribution, so we will generally use $(x,C)$ in place of
↪ $1,\ldots,d$ when addressing a specific position in a specific choice system $q$.


Our definition of a choice system differs from Falmagne's \cite{falmagne1978representation,barbera1986falmagne} related definition of a {\it
↪ system of choice probabilities} $\{P_{x,C}\}_{\forall C \subseteq \mathcal X, \forall x \in C}$. Our definition of a system explicitly includes the
↪ probability distribution over sets, $w(C)$, $\forall C \in \mathcal X$, which Falmagne's concept does not. Falmagne's definition can be thought of
↪ as studying choices separated from an exogenously defined collection of choice sets. Further, because our choice systems are built to span only
↪ observed $C \in \mathcal{C}$, under our definition all choice systems have a positive set probability $w(C)>0$ for all $C \in \mathcal{C}$.
↪ Without the assumption that $w(C)>0$ for all $C \in \mathcal{C}$, the worst case error of any test of IIA would be trivially large, since it has no
↪ chance of measuring a violation of IIA within a set $C$ that it can not observe.

\subsection{Choice systems and IIA}
The Independence of Irrelevant Alternatives (IIA) assumption constrains the probabilities $P_{x,C}$ to satisfy the following ratio for any $C \subset
↪ \mathcal{X}$:
\begin{align*}
\dfrac{P_{x,\lbrace x, y \rbrace \cup C}}{P_{y,\lbrace x, y \rbrace \cup C}} = \dfrac{P_{x,\lbrace x, y \rbrace}}{P_{y,\lbrace x, y \rbrace}}.
\end{align*}
As derived by Luce in \cite{luce1959individual}, the IIA assumption consequently admits what Luce called a ratio representation,
\begin{align*}
P_{x,C} = \frac{\gamma_x}{\sum_{z \in C} \gamma_z},
\end{align*}
for some $\gamma \in \mathbb{R}_+^n$. Clearly, $\gamma$ is scale invariant. Setting a scale such that $\sum_{z \in \mathcal{X}} \gamma_z = 1$
↪ gives the interpretation that $\gamma_z = P_{z, \mathcal{X}}$.
That is, this interpretation says that an IIA choice system can be described by just knowing $w(C)$, $\forall  C \in \mathcal{C}$, and
↪ $P_{x,\mathcal{X}}$, $\forall x \in \mathcal X$.

We let $\Piia \subset \Pall$ denote the collection of all choice systems that satisfy IIA. Essentially, then, where $\Pall$ is the space of all
↪ $d$-dimensional discrete distributions, $\Piia$ is a family of (non-linearly) constrained $d$-dimensional discrete distributions. We emphasize that
↪ $\Piia$ is non-convex, and that the convex hull of $\Piia$ is dense in $\Pall$; the latter fact we show in the appendix (Fact~\ref{cvx_hull_dense}).
```

Expanded PaperInterface specification

```
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q : SeshadriUgander2020IIATesting.ChoiceSystem F) (ε : ℝ),
  0 < ε → ∃ (r : SeshadriUgander2020IIATesting.ChoiceSystem F), r.InFiniteConvexHullIIA ∧ q.totalVariation r < ε
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.InFiniteConvexHullIIA
- SeshadriUgander2020IIATesting.ChoiceSystem.totalVariation

Lean-elaborated claim roles

1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. parameter: SeshadriUgander2020IIATesting.ChoiceSystem F
3. parameter: ℝ
4. assumption: 0 < ε
5. conclusion: ∃ (r : SeshadriUgander2020IIATesting.ChoiceSystem F), r.InFiniteConvexHullIIA ∧ q.totalVariation r < ε

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.fact_convex_hull_dense

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

Reason: The source says that the finite convex hull of IIA choice systems is dense in all choice systems. The target states the equivalent epsilon-form density assertion in total variation: every choice system has, at every positive tolerance, a finite mixture of IIA components within that tolerance.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 6

Source locator: cited publication, lines 790-804

Verbatim source input

```
\begin{lemma}\label{lemma:iia_invariance}
		Given two choice systems $q_1$ and $q_2$ defined over the same collection $\mathcal{C}$ such that
		$$
		\sum_{j \in C} {q_1}_{(j,C)} = \sum_{j \in C} {q_2}_{(j,C)} = \mu_C, \ \
		\forall C \in \mathcal{C},$$ and
		$$\sum_{\substack{C \in \mathcal{C}\\ C \ni i}} {q_1}_{(i,C)} = \sum_{\substack{C \in \mathcal{C}\\ C \ni i}} {q_2}_{(i,C)} = \alpha_i, \ \
		\forall i \in \mathcal{X},$$
		then,
		$$
		\argmin_{p \in \Piia}
		D_{\text{KL}}(q_1||p) =
		\argmin_{p \in \Piia}
		D_{\text{KL}}(q_2||p).
		$$
	\end{lemma}
```

[Next verbatim source excerpt]

```
	\subsection{Choice systems}\label{subsection:choice_system}
	Let $\mathcal{X}$ be a finite set of $n$ \textit{items} with generic element $x \in \mathcal{X}$, and let $\mathcal{C} \subseteq 2^\mathcal{X}$
	↪ denote the observed space, a collection of $m \geq n$ unique unordered subsets $C \subseteq \mathcal{X}$ of size 2 or greater. A decision
	↪ maker who is presented with a \textit{choice set} $C \in \mathcal{C}$ chooses one item from the set. We let $P_{x,C}$ denote the probability
	↪ that $x$ is chosen from $C$, $\forall x \in C$, and let $w(C)$ denote the probability of seeing choice set $C$, $\forall C \in \mathcal{C}$. The
	↪ object $\mathcal{C}$ should be treated as a sample frame---the regime of sets of interest, or the regime of sets for which a notion of error is
	↪ important---determined ahead of time (as opposed to being defined in terms of the sets so far observed in some dataset). That is, we are
	↪ operating in a fixed-design setting, where $\mathcal{C}$ does not depend on, e.g., prior observations.

	Our test is focused on a given dataset $\mathcal{D}_N = \lbrace (x_j, C_j) \rbrace_{j=1}^N$ that results from a decision maker making choices: a
	↪ datapoint $j$ represents a decision scenario, and contains $C_j$, the choice set provided in that decision, and $x_j \in C_j$, the item chosen from
	↪ that choice set. We assume each datapoint is an independent sample from the joint distribution over choices sets and choices:
	\begin{align*}
	Pr[(x_j, C_j) = (x, C)] = w(C) \cdot P_{x,C}, \ \ \forall j.
	\end{align*}
	That is, a datapoint is a sample from some discrete distribution over $d$ possible (choice, choice set) pairs, where $d = \sum_{C \in \mathcal{C}}
	↪ |C|$. We refer to this $d$-dimensional discrete distribution as a \textit{choice system} and denote it using $q$.
	Let $\Pall$ denote the collection of all choice systems on the collection $\mathcal{C}$, i.e., the collection of all $d$-dimensional discrete
	↪ distributions. The (choice, choice set) tuples $(x,C)$ can be used to intuitively index this distribution, so we will generally use $(x,C)$ in place of
	↪ $1,\ldots,d$ when addressing a specific position in a specific choice system $q$.


	Our definition of a choice system differs from Falmagne's \cite{falmagne1978representation,barbera1986falmagne} related definition of a {\it
	↪ system of choice probabilities} $\{P_{x,C}\}_{\forall C \subseteq \mathcal X, \forall x \in C}$. Our definition of a system explicitly includes the
	↪ probability distribution over sets, $w(C)$, $\forall C \in \mathcal X$, which Falmagne's concept does not. Falmagne's definition can be thought of
	↪ as studying choices separated from an exogenously defined collection of choice sets. Further, because our choice systems are built to span only
	↪ observed $C \in \mathcal{C}$, under our definition all choice systems have a positive set probability $w(C)>0$ for all $C \in \mathcal{C}$.
	↪ Without the assumption that $w(C)>0$ for all $C \in \mathcal{C}$, the worst case error of any test of IIA would be trivially large, since it has no
	↪ chance of measuring a violation of IIA within a set $C$ that it can not observe.

	\subsection{Choice systems and IIA}
	The Independence of Irrelevant Alternatives (IIA) assumption constrains the probabilities $P_{x,C}$ to satisfy the following ratio for any $C \subset
	↪ \mathcal{X}$:
	\begin{align*}
	\dfrac{P_{x,\lbrace x, y \rbrace \cup C}}{P_{y,\lbrace x, y \rbrace \cup C}} = \dfrac{P_{x,\lbrace x, y \rbrace}}{P_{y,\lbrace x, y \rbrace}}.
	\end{align*}
	As derived by Luce in \cite{luce1959individual}, the IIA assumption consequently admits what Luce called a ratio representation,
	\begin{align*}
	P_{x,C} = \frac{\gamma_x}{\sum_{z \in C} \gamma_z},
	\end{align*}
	for some $\gamma \in \mathbb{R}_+^n$. Clearly, $\gamma$ is scale invariant. Setting a scale such that $\sum_{z \in \mathcal{X}} \gamma_z = 1$
	↪ gives the interpretation that $\gamma_z = P_{z, \mathcal{X}}$.
	That is, this interpretation says that an IIA choice system can be described by just knowing $w(C)$, $\forall  C \in \mathcal{C}$, and
	↪ $P_{x,\mathcal{X}}$, $\forall x \in \mathcal X$.

	We let $\Piia \subset \Pall$ denote the collection of all choice systems that satisfy IIA. Essentially, then, where $\Pall$ is the space of all
	↪ $d$-dimensional discrete distributions, $\Piia$ is a family of (non-linearly) constrained $d$-dimensional discrete distributions. We emphasize that
	↪ $\Piia$ is non-convex, and that the convex hull of $\Piia$ is dense in $\Pall$; the latter fact we show in the appendix (Fact~\ref{cvx_hull_dense}).
```

[Next verbatim source excerpt]

```
		\begin{proof}
			From the definition of KL divergence and IIA, and removing terms that don't affect the argument, we have
			\begin{align*}
			\text{arg} \inf_{p \in \Piia} D_{KL}(q||p)
			&= \text{arg} \inf_{p \in \Piia} \sum_{C \in \mathcal{C}} \sum_{j \in C} q_{(j,C)} \log\Big(\frac{q_{(j,C)}}{p_{(j,C)}}\Big)\\
			&= \text{arg} \inf_{p \in \Piia} \sum_{C \in \mathcal{C}} \sum_{j \in C} q_{(j,C)} \log(q_{(j,C)}) - \sum_{C \in \mathcal{C}} \sum_{j \in C}
			↪ q_{(j,C)} \log(p_{(j,C)})\\
			&=\text{arg} \inf_{w \in \Delta_m, \theta \in \mathbb{R}^n}
			- \sum_{C \in \mathcal{C}} \sum_{j \in C} q_{(j,C)} \log\Big(\frac{w_C \exp(\theta_j)}{\sum_{k \in C} \exp(\theta_k)}\Big).
			\end{align*}
			We can further rewrite this expression using properties of logarithms and rearranging terms:
			\begin{align*}
			\text{arg} \inf_{p \in \Piia} D_{KL}(q||p)
			&=\text{arg} \inf_{w \in \Delta_m, \theta \in \mathbb{R}^n}
			- \sum_{C \in \mathcal{C}} \log(w_C)\sum_{j \in C} q_{(j,C)} - \sum_{C \in \mathcal{C}} \sum_{j \in C} q_{(j,C)}
			↪ \log\Big(\frac{\exp(\theta_j)}{\sum_{k \in C} \exp(\theta_k)}\Big) \\
			&=\text{arg} \inf_{w \in \Delta_m, \theta \in \mathbb{R}^n}
			- \sum_{C \in \mathcal{C}} \sum_{k \in C} q_{(k,C)}\log(w_C) - \sum_{C \in \mathcal{C}} \sum_{j \in C} q_{(j,C)}
			↪ \log\Big(\frac{\exp(\theta_j)}{\sum_{k \in C} \exp(\theta_k)}\Big).
```

\end{align*}
Next we notice that we can optimize over w by applying Fact \ref{max_entropy} (see below), that is, $w^\star = \sum_{k \in C}
↪ q_{(k,C)}$, and a few additional steps of algebra:
\begin{align*}
\text{arg} \inf_{p \in \Piia} D_{KL}(q||p)
&=\text{arg} \inf_{\theta \in \mathbb{R}^n}
- \sum_{C \in \mathcal{C}} \sum_{j \in C} q_{(j,C)} \log\Big(\frac{\exp(\theta_j)}{\sum_{k \in C} \exp(\theta_k)}\Big)\\
&= \text{arg} \inf_{\theta \in \mathbb{R}^n} - \sum_{C \in \mathcal{C}} \sum_{j \in C} \theta_j q_{(j,C)} + \sum_{C \in \mathcal{C}}
↪ \sum_{j \in C} q_{(j,C)} \log(\sum_{k \in C} \exp(\theta_k))\\
&= \text{arg} \inf_{\theta \in \mathbb{R}^n} - \sum_{j \in \mathcal{X}} \theta_j \sum_{\substack{C \in \mathcal{C}\\ C \ni j}} q_{(j,C)}
↪ + \sum_{C \in \mathcal{C}} \log(\sum_{k \in C} \exp(\theta_k)) \sum_{j \in C} q_{(j,C)}.
\end{align*}

Now, we see that,
\begin{align*}
\text{arg} \inf_{p \in \Piia} D_{KL}(q_1||p)
&= \text{arg} \inf_{\theta \in \mathbb{R}^n} - \sum_{j \in \mathcal{X}} \theta_j \alpha_j + \sum_{C \in \mathcal{C}} \mu_C \log(\sum_{k
↪ \in C} \exp(\theta_k)) \\
&= \text{arg} \inf_{p \in \Piia} D_{KL}(q_2||p).
\end{align*}
This result concludes the proof.

Expanded PaperInterface specification

∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (q₁ q₂ : SeshadriUgander2020IIATesting.ChoiceSystem F),
  (∀ (C : F.SetId), q₁.setMass C = q₂.setMass C) →
    (∀ (x : F.Item), q₁.itemMass x = q₂.itemMass x) → q₁.iiaProjection = q₂.iiaProjection

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceSystem
- SeshadriUgander2020IIATesting.ChoiceSystem.iiaProjection
- SeshadriUgander2020IIATesting.ChoiceSystem.itemMass
- SeshadriUgander2020IIATesting.ChoiceSystem.setMass

Lean-elaborated claim roles

1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. parameter: SeshadriUgander2020IIATesting.ChoiceSystem F
3. parameter: SeshadriUgander2020IIATesting.ChoiceSystem F
4. assumption: ∀ (C : F.SetId), q₁.setMass C = q₂.setMass C
5. assumption: ∀ (x : F.Item), q₁.itemMass x = q₂.itemMass x
6. conclusion: q₁.iiaProjection = q₂.iiaProjection

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.lemma_iia_projection_invariance

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

Reason: The two Lean hypotheses are exactly equality of the set marginals and equality of the item marginals of q1 and q2. The conclusion identifies their complete sets of KL-minimizing IIA choice systems; the positive set-mass restriction is the source IIA model's logarithm domain, so this is the stated argmin-set invariance.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 7

Source locator: cited publication, lines 840-848

Verbatim source input

```
\begin{fact}\label{max_entropy}
        For any $q \in \Delta_s$,
        \begin{align}
        \inf_{p \in \Delta_s}
        -\sum_{i=1}^s q_i \log(p_i)
        =  H(q),
        \end{align}
        where $H(q)$ denotes the entropy of $q$.
    \end{fact}
```

[Next verbatim source excerpt]

```
    \begin{proof}
        \begin{align*}
        \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) &= \inf_{p \in \Delta_s} -\sum_{i=1}^s q_i \log(p_i) + \sum_{i=1}^s q_i \log(q_i) -
        ↪   \sum_{i=1}^s q_i \log(q_i)\\
        &= \inf_{p \in \Delta_s} \sum_{i=1}^s q_i \log\Big(\frac{q_i}{p_i}\Big)+ \sum_{i=1}^s q_i \log\Big(\frac{1}{q_i}\Big)\\
        &= \inf_{p \in \Delta_s} D_{KL}(q||p) + H(q)\\
        &=H(q),
        \end{align*}
        where the last line follows from the non-negativity of KL divergence, and it being zero if and only if $p = q$.
```

Expanded PaperInterface specification

```
∀ {α : Type u_1} [inst : Fintype α] [inst_1 : DecidableEq α] (q : PMF α),
  sInf (AppliedModelingLib.finiteCrossEntropyExtendedRange q) = ↑(AppliedModelingLib.finiteEntropy q)
```

Retained governing declarations

- AppliedModelingLib.finiteCrossEntropyExtendedRange
- AppliedModelingLib.finiteEntropy

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype α
3. parameter: DecidableEq α
4. parameter: PMF α
5. conclusion: sInf (AppliedModelingLib.finiteCrossEntropyExtendedRange q) = ↑(AppliedModelingLib.finiteEntropy q)

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.fact_entropy_cross_entropy

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

Reason: For every finite probability mass function q, the target is the source finite-simplex equality between the infimum of cross entropy and entropy. Its extended-real cross-entropy presentation preserves the source equality at zero-probability boundary points.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 8

Source locator: cited publication, lines 867-869

Verbatim source input

```
\begin{lemma}\label{lemma:bipartite_cycle_decomposition}
    Given a bipartite graph $G = (V_1, V_2, E)$ with $n_1 = |V_1|$ and $n_2 = |V_2|$ vertices in each part (wlog $n_2 \geq n_1$) and $d = |E|$
    ↪ edges, there exists a cycle decomposition of the edge set into cycles of size at most $2 \floor{2 \log_2(n_1)}$ with at most
    ↪ $\min\{2n_1 + n_2, 4n_1 + n_2^\text{(odd)}\}$ edges remaining, where $n_2^\text{(odd)} \leq n_2$ represents the number of
    ↪ vertices of odd degree on the second vertex set.
\end{lemma}
```

[Next verbatim source excerpt]

```
    The proof follows the same general structure as that of Lemma \ref{lemma:general_cycle_decomposition}, but modifies a few steps to
    ↪ directly incorporate properties of the bipartite graph, and therefore produces results more specific to bipartite graphs. We begin by
    ↪ stating a procedure to construct a cycle decomposition on an arbitrary bipartite graph $G$ with some edges remaining.

    First, repeatedly remove vertices, along with their incident edges, of degree 2 or less from $V_1$ and of degree 1 or less from $V_2$.
    ↪ Second, perform a breadth-first search (BFS) starting from any remaining vertex in $V_1$ until a cycle $\sigma_1$ is found. Add the
    ↪ cycle to the collection of cycles $\sigma$, and remove the edges from $G$. Repeat these two steps, adding cycles $\sigma_i$ to
    ↪ $\sigma$ until no edges remain in $G$.

    Now, we can analyze this procedure. Note that the only edges removed from $G$ that are also not added to $\sigma$ are from vertices of
    ↪ degree 2 or less from $V_1$ and from vertices of degree 1 or less from $V_2$. This fact allows us to upper bound the number of edges
    ↪ not contained in the cycle decomposition by $2n_1 + n_2$.

    Now, consider any instance of the second step of the procedure above. Since all vertices of degree less than 2 have been removed from
    ↪ $V_1$, $V_1$ has vertices of minimum degree 3. Similarly, since all vertices of degree less than 1 have been removed from $V_2$,
    ↪ $V_2$ only has vertices of minimum degree 2. Thus, the arbitrary vertex in $V_1$ at the start of the BFS is connected to at least three
    ↪ vertices in $V_2$. Those three verticies, are in turn connected to at least one new vertex each in $V_1$ if a cycle is not found, all of
    ↪ which are in turn connected to two new vertices each in $V_2$ if a cycle is not found, and so on. Thus, as long as we don't find a cycle,
    ↪ the BFS tree grows at least as a complete binary tree does every time we go to $V_2$. Since there are no leaf nodes in this graph (no
    ↪ vertex has degree 1), a cycle must eventually be found. The maximum depth of the BFS tree is then at most $\floor{2 \log_2(n_1)}$,
    ↪ twice the depth of a complete binary tree since that is the maximum depth before all nodes in $V_1$ are visited and a cycle has to be
    ↪ found. The cycle can then be at most twice the depth of the BFS tree, which upper bounds the size of the cycle as $2\floor{2
    ↪ \log_2(n_1)}$. Thus, the key difference in this step of the proof when contrasted with that of Lemma
    ↪ \ref{lemma:general_cycle_decomposition} is using $V_2$ as a bridge back to $V_1$ and using $V_1$ only to exponentially grow the
    ↪ BFS tree. The analytical segmentation of the vertices then allows a bound on the size of the cycle explicitly in terms of the number of
    ↪ vertices of $V_1$, which is useful when the vertices are highly imbalanced in the two parts.

    Since both steps can always finish as long as $G$ has edges and the second always strictly reduces the number of edges of $G$, the
    ↪ algorithm eventually terminates and produces a cycle decomposition with cycles of size at most $2\floor{2 \log_2(n_1)}$ with at most
    ↪ $2n_1 + n_2$ edges that remain.

    We close by furnishing the proof for the slightly sharper bound on the edges that remain that is stated in the Lemma. Consider first that
    ↪ removing the edges corresponding to a cycle in a graph modifies the degree of every vertex by an even number, and hence does not
    ↪ change the parity of the vertex's degree. Next, consider that during the first step, edges are removed due to the removal of vertices in
    ↪ $V_2$ only if those vertices are degree 1. Thus, vertices in $V_2$ must either begin with odd parity in order for one of their edges to
    ↪ not contribute to the cycle decomposition, or they must be changed to odd parity due to edges removed from the removal of vertices
    ↪ from $V_1$ in the first step. Since vertices removed from $V_1$ in the first step can have degree at most 2, their removal changes the
    ↪ parity of at most $2n_1$ vertices in $V_2$. This argument yields an alternate upper bound of edges removed that do not contribute to
    ↪ the cycle decomposition, $2n_1 + 2n_1 + n_2^\text{(odd)}$, and concludes the proof.
```

Expanded PaperInterface specification

```
∀ {V : Type u_1} [inst : Fintype V] [inst_1 : DecidableEq V] (G : SimpleGraph V) [DecidableRel G.Adj]
  (left right : Set V),
  G.IsBipartiteWith left right →
    ∃ (P : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G (2 * ⌊2 * Real.logb 2 ↑left.ncard⌋₊)),
      AppliedModelingLib.Foundations.Graph.edgeCount G - P.usedEdges.card ≤
        min (2 * left.ncard + right.ncard)
          (4 * left.ncard + AppliedModelingLib.Foundations.Graph.oddDegreeCount G right)
```

Retained governing declarations

- AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking
- AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking.usedEdges
- AppliedModelingLib.Foundations.Graph.edgeCount
- AppliedModelingLib.Foundations.Graph.oddDegreeCount

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype V
3. parameter: DecidableEq V
4. parameter: SimpleGraph V
5. parameter: DecidableRel G.Adj
6. parameter: Set V
7. parameter: Set V
8. assumption: G.IsBipartiteWith left right
9. conclusion: ∃ (P : AppliedModelingLib.Foundations.Graph.PartialSimpleCyclePacking G (2 * ⌊2 * Real.logb 2 ↑left.ncard⌋₊)),
   AppliedModelingLib.Foundations.Graph.edgeCount G - P.usedEdges.card ≤
     min (2 * left.ncard + right.ncard) (4 * left.ncard + AppliedModelingLib.Foundations.Graph.oddDegreeCount G right)

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.lemma_bipartite_cycle_decomposition

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

Reason: The target expresses exactly the source bipartite cycle-decomposition conclusion: a packing with cycle cap 2 floor(2 log_2 |V_1|) and uncovered-edge bound min(2|V_1|+|V_2|, 4|V_1|+n_2^(odd)). Omitting the source's w.l.o.g. size order establishes the same claim on a stronger domain.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 9

Source locator: cited publication, lines 884-890

Verbatim source input

```
\begin{lemma}\label{lemma:Gc_cycle_decomposition}
    Any Eulerian comparison incidence graph $G_\mathcal{C}$, defined in Section \ref{subsection:comparison_graph}, has a cycle
    ↪ decomposition $\sigma$ where all but at most $\min(2n+m, 4n)$ edges (i.e., at most $\max(d-\min(2n+m, 4n), 0)$ edges) contribute
    ↪ to cycles of size at most $2\floor{2 \log_2(n)}$ and the rest (at most $\min(2n+m, 4n)$) contribute to cycles of size at most $2n$. We
    ↪ then have the following bounds on $\mu(\sigma)$ and $\alpha(\sigma)$:
    \begin{align*}
    \mu(\sigma) \leq \min \Big\{\Big(\frac{d}{d - 4n + 8\log_2 (n)}\Big)4\log_2 (n), 2n\Big\} \\
    \alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d}, 2n\Big\}.
    \end{align*}
\end{lemma}
```

[Next verbatim source excerpt]

```
\subsection{Comparison incidence graphs}\label{subsection:comparison_graph}
To represent the comparison structure imposed by $\mathcal{C}$, we consider an undirected bipartite graph $G_\mathcal{C}$ with $n$ nodes for
↪ each item in $\mathcal{X}$ and $m$ nodes for each set in $\mathcal{C}$. Edges in this graph are drawn from the item nodes to the set nodes
↪ to indicate membership. That is, the nodes corresponding to each $C \in \mathcal{C}$ have edges extending to the items contained in $C$. This
↪ bipartite graph can also be thought of as representing a hypergraph with $n$ nodes, one for each item in $\mathcal{X}$, and $m$ hyperedges,
↪ one for each set in $\mathcal{C}$. The graph $G_\mathcal{C}$ is then the incidence graph of this hypergraph, and contain $d$ edges. As a
↪ result we call $G_\mathcal{C}$ the {\it comparison incidence graph}, to avoid confusion with other objects called comparison graphs in the
↪ study of pairwise comparisons \cite{ford1957solution,negahban2016rank}.

Throughout this work, we will assume that $G_\mathcal{C}$ is connected, meaning that there is a path in the bipartite graph from every item $i$ to
↪ every item $j$.

This assumption corresponds to the necessary and sufficient condition for inference of an IIA choice system first established for pairs by
↪ Ford~\cite{ford1957solution} and later more generally by Hunter~\cite{hunter2004mm}. Moreover, we note that the requirement that $m \geq
↪ n$, stated earlier, is a sufficient condition on $\mathcal{C}$ to guarantee that there are cycles in $G_\mathcal{C}$\footnote{The necessary and
↪ sufficient condition to guarentee cycles in a connected $G_\mathcal{C}$ is $d \geq m + n$. Though our analysis still applies in this regime, we
↪ prefer the stronger condition $m \geq n$ to produce shorter analyses and cleaner expressions at the cost of sharpness when $m < n$.}. As we
↪ shall see, cycles are critically important: a $G_\mathcal{C}$ without cycles corresponds to a collection $\mathcal{C}$ for which all choice
↪ systems are IIA, rendering a test for IIA meaningless.

We call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}, and denote a cycle decomposition by
↪ $\sigma$. We define two functions of $\sigma$ that appear in the lower bound: $\mu(\sigma) = \frac{d}{|\sigma|}$, the \textit{average cycle
↪ length} of cycles in the decomposition and $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$, the \textit{cycle dispersion
↪ index}\footnote{Although the index of dispersion traditionally refers to a ratio of variance and mean, we abuse the term slightly here to refer to
↪ a ratio of the cycle lengths' second moment and the cycle lengths' mean.} of cycles in the decomposition. We will elaborate on these terms
↪ further in Section \ref{sec:bounding}.

For simplicity of presentation of the results and proofs, for the remainder of this work we only consider the special case of collections $\mathcal{C}$
↪ for which $|C|$ is even, $\forall C \in \mathcal{C}$, and where every item appears an even number of times across the subsets in
↪ $\mathcal{C}$. Under this restriction, we note that every node in $G_\mathcal{C}$ (on both sides of the bipartite graph) has even degree. As a
↪ consequence, $G_\mathcal{C}$ is Eulerian, which will be critical for our analysis.
We note that a graph $G_\mathcal{C}$ with odd degrees can be made Eulerian by removing simple paths between odd-degree nodes. While we do
↪ not explicitly extend our lower bounds beyond $\mathcal C$ for which $G_\mathcal{C}$ is Eulerian, a modest exercise in bookkeeping relaxes
↪ the requirement of even set sizes and even item appearances.
```

[Next verbatim source excerpt]

```
    \begin{proof}
        Apply Lemma \ref{lemma:bipartite_cycle_decomposition} to $G_\mathcal{C}$, with $n_1 = n$ and $n_2 = m$. Since $G_\mathcal{C}$ is
        ↪ Eulerian, $n_2^\text{(odd)} = 0$, and so we know that $G_C$ has a cycle decomposition with cycles of size at most $2 \floor{2
        ↪ \log(n_1)}$ with at most $\min(2n + m, 4n)$ edges that remain. Again since $G_\mathcal{C}$ is Eulerian, and removing cycles from
        ↪ $G_\mathcal{C}$ does not change this fact, the remaining $\min(2n + m, 4n)$ edges can always be decomposed into cycles of size at
        ↪ most $2n$. The latter bound on the cycle size follows from the application of the pigeonhole principle described in Section
        ↪ $\ref{sec:relax}$. When combined, these steps form the $sigma$ that possesses the properties stated in the Lemma.

        Naturally, the short cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are only guaranteed if $d$ exceeds $\min(2n + m,
        ↪ 4n)$, else we use the extremal bound on cycle size of $2n$ for all edges in $d$. Note that $2\floor{2 \log_2(n)} \leq 2n$ for all $n \in
        ↪ \mathbb{N}$, so the cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are always smaller than those guaranteed by the
        ↪ extremal bound. We then have the following lower bound on $|\sigma|$, the number of cycles in the cycle decomposition:
        \begin{align*}
        |\sigma| \geq \max \Big\{\frac{d - 2n - \min(2n, m)}{2\floor{2 \log_2(n)}} + \frac{2n + \min(2n, m)}{2n}, \frac{d}{2n}\Big\}.
        \end{align*}
        We replace $\min(2n,m)$ with $2n$ to weaken the above bound at the benefit of making the expression much simpler, especially for its
        ↪ use bounding $\mu(\sigma)$ and $\alpha(\sigma)$. We also weaken $2\floor{2 \log_2(n)}$ to $4 \log_2(n)$ for the same purpose. For
        ↪ the sharpest results, we invite the reader to carry the original bound over to $\mu(\sigma)$ and $\alpha(\sigma)$ in a manner similar
        ↪ to what we do below. We then have:
        \begin{align*}
        |\sigma| \geq \max \Big\{\frac{d - 4n}{4\log_2 (n)} + 2, \frac{d}{2n}\Big\}.
        \end{align*}
        Since $\mu(\sigma) = \frac{d}{|\sigma|}$, the preceding bound immediately yields an upper bound on $\mu(\sigma)$:
        \begin{align*}
        \mu(\sigma) \leq \frac{d}{\max \Big\{\frac{d - 4n}{4\log_2(n)} + 4, \frac{d}{2n}\Big\}} = \min \Big\{\Big(\frac{d}{d - 4n + 8\log_2
        ↪ (n)}\Big)4\log_2 (n), 2n\Big\}.
        \end{align*}
        Now, we bound $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$. Clearly, $\alpha(\sigma)$ grows with the size of
        ↪ the cycles in $\sigma$, so we may upper bound it by considering the largest size the cycles in $\sigma$ could be. Recall that
        ↪ $2\floor{2 \log_2(n)} \leq 2n$ for all $n \in \mathbb{N}$, so the cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are
        ↪ always smaller than those guaranteed by the extremal bound of $2n$. Thus, the more size-$2n$ cycles in our cycle decomposition, the
        ↪ higher $\alpha(\sigma)$. Since we can have at most $4n$ edges contribute to cycles of size at most $2n$, if $d \leq 4n$, then we have:
        \begin{align*}
        \alpha(\sigma) \leq \frac{1}{d} \frac{d}{2n} (2n)^2 = 2n.
        \end{align*}
        Otherwise, we have:
        \begin{align*}
```

```
\alpha(\sigma) \leq \frac{1}{d} \Big(\frac{d - 4n}{2\floor{2 \log_2(n)}}(2\floor{2 \log_2(n)})^2 + \frac{4n}{2n}(2n)^2 \Big) \leq 4
↪  \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d},
\end{align*}
where the first inequality follows from computing the formula with all $4n$ edges in cycles at their maximum size of $2n$ and the
↪  remaining edges in cycles at their maximum size of $2\floor{2 \log_2(n)}$. The second inequality comes from rearranging terms and
↪  weakening $2\floor{2 \log_2(n)}$ to $4 \log_2(n)$. Putting the two cases together, we have:
\begin{align*}
\alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d}, 2n\Big\}.
\end{align*}
\end{proof}
```

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

Every Eulerian comparison-incidence graph with at least four items and positive rational-branch denominator has the source two-tier cycle decomposition and the corrected displayed mean and dispersion bounds.

Archival source anchor: cited publication, lines 884-890

Expanded PaperInterface specification

```
∀ (F : SeshadriUgander2020IIATesting.ChoiceFrame),
  F.Eulerian →
    4 ≤ Fintype.card F.Item →
      0 < ↑F.incidenceCount - 4 * ↑(Fintype.card F.Item) + 2 * (4 * Real.logb 2 ↑(Fintype.card F.Item)) →
        ∃ (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F),
          D.cycleMean ≤ SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound ∧
            SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length ≤
              SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Eulerian
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound
- SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean
- SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion

Lean-elaborated claim roles

```
1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. assumption: F.Eulerian
3. assumption: 4 ≤ Fintype.card F.Item
4. assumption: 0 < ↑F.incidenceCount - 4 * ↑(Fintype.card F.Item) + 2 * (4 * Real.logb 2 ↑(Fintype.card F.Item))
5. conclusion: ∃ (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F),
  D.cycleMean ≤ SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryMeanBound ∧
    SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion F.incidenceCount D.length ≤
      SeshadriUgander2020IIATesting.ChoiceFrame.sourceCorollaryDispersionBound
```

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.lemma_comparison_incidence_decomposition_corrected

Recorded source-to-Spec screening

Verdict: matches_approved_corrected_target
Reason: The target constructs the source comparison-incidence cycle decomposition and states its displayed mean and dispersion bounds with d as incidenceCount and n as item count. It explicitly includes the approved n >= 4 real-log range and positive denominator condition for the rational branch.
Reviewer check: ☐ Matches formalized target
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 10

Source locator: cited publication, lines 919-925

Verbatim source input

```
\begin{cor}[]\label{cor:all_subsets_cycle_decomposition}
        For the comparison graph of all subsets of even size described in Section \ref{subsection:all}, $\mu(\sigma)$ and $\alpha(\sigma)$ are
        ↪ bounded as follows:
        \begin{align*}
        \mu(\sigma) &\leq 5\log_2 (n), \\
        \alpha(\sigma) &\leq 5\log_2 (n).
        \end{align*}
    \end{cor}
```

[Next verbatim source excerpt]

```
    \subsection{All subsets}
    \label{subsection:all}
    IIA is defined as a property of the complete choice system of $n$ items, and hence, any true test of IIA must encompass all the subsets of an item
    ↪ universe. We thus focus first on the results of our lower bound on this problem instance. We again consider the slightly easier setting amenable
    ↪ to our analysis where we only study (all) subsets of even size. Over the unique even subsets of a choice system, a simple calculation reveals that
    ↪ every item then appears the following number of times,
    \begin{align*}
    \sum_{k=1}^{\floor{n/2}} \binom{n-1}{2k-1} = 2^{n-2},
    \end{align*}
    which is always an even number. Hence, the resulting comparison graph is Eulerian, and our lower bound applies with $d=n2^{n-2}$. Given the
    ↪ sheer size of $d$, we do not lose much in the flavor of our statement by using the upper bounds for $\alpha(\sigma)$ and $\mu(\sigma)$ from
    ↪ Lemma \ref{lemma:Gc_cycle_decomposition}. In fact, since $d$ is exponentially large in $n$, the expressions for both $\alpha(\sigma)$ and
    ↪ $\mu(\sigma)$ can be upper bounded by $c \log(n)$ where $c$ is some constant\footnote{This argument is detailed in Corollary
    ↪ \ref{cor:all_subsets_cycle_decomposition}, a corollary of Lemma~\ref{lemma:Gc_cycle_decomposition}, in the Appendix.}. We then have the
    ↪ following lower bound for the setting of all subsets of even size, and consequently, a lower bound for the setting of all
    ↪ subsets\footnote{Conveniently, $d=n2^{n-1}$ for the case of all subsets (even or odd sizes), and we can use this value in our lower bound as it
    ↪ is within a constant factor of $d=n2^{n-2}$ for the all even subsets case.}:
```

[Next verbatim source excerpt]

```
        \begin{lemma}\label{lemma:Gc_cycle_decomposition}
            Any Eulerian comparison incidence graph $G_\mathcal{C}$, defined in Section \ref{subsection:comparison_graph}, has a cycle
            ↪ decomposition $\sigma$ where all but at most $\min(2n+m, 4n)$ edges (i.e., at most $\max(d-\min(2n+m, 4n), 0)$ edges) contribute
            ↪ to cycles of size at most $2\floor{2 \log_2(n)}$ and the rest (at most $\min(2n+m, 4n)$) contribute to cycles of size at most $2n$. We
            ↪ then have the following bounds on $\mu(\sigma)$ and $\alpha(\sigma)$:
            \begin{align*}
            \mu(\sigma) \leq \min \Big\{\Big(\frac{d}{d - 4n + 8\log_2 (n)}\Big)4\log_2 (n), 2n\Big\} \\
            \alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d}, 2n\Big\}.
            \end{align*}
        \end{lemma}
        \begin{proof}
            Apply Lemma \ref{lemma:bipartite_cycle_decomposition} to $G_\mathcal{C}$, with $n_1 = n$ and $n_2 = m$. Since $G_\mathcal{C}$ is
            ↪ Eulerian, $n_2^\text{(odd)} = 0$, and so we know that $G_C$ has a cycle decomposition with cycles of size at most $2 \floor{2
            ↪ \log(n_1)}$ with at most $\min(2n + m, 4n)$ edges that remain. Again since $G_\mathcal{C}$ is Eulerian, and removing cycles from
            ↪ $G_\mathcal{C}$ does not change this fact, the remaining $\min(2n + m, 4n)$ edges can always be decomposed into cycles of size at
            ↪ most $2n$. The latter bound on the cycle size follows from the application of the pigeonhole principle described in Section
            ↪ $\ref{sec:relax}$. When combined, these steps form the $sigma$ that possesses the properties stated in the Lemma.

            Naturally, the short cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are only guaranteed if $d$ exceeds $\min(2n + m,
            ↪ 4n)$, else we use the extremal bound on cycle size of $2n$ for all edges in $d$. Note that $2\floor{2 \log_2(n)} \leq 2n$ for all $n \in
            ↪ \mathbb{N}$, so the cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are always smaller than those guaranteed by the
            ↪ extremal bound. We then have the following lower bound on $|\sigma|$, the number of cycles in the cycle decomposition:
            \begin{align*}
            |\sigma| \geq \max \Big\{\frac{d - 2n - \min(2n, m)}{2\floor{2 \log_2(n)}} + \frac{2n + \min(2n, m)}{2n}, \frac{d}{2n}\Big\}.
            \end{align*}
            We replace $\min(2n,m)$ with $2n$ to weaken the above bound at the benefit of making the expression much simpler, especially for its
            ↪ use bounding $\mu(\sigma)$ and $\alpha(\sigma)$. We also weaken $2\floor{2 \log_2(n)}$ to $4 \log_2(n)$ for the same purpose. For
            ↪ the sharpest results, we invite the reader to carry the original bound over to $\mu(\sigma)$ and $\alpha(\sigma)$ in a manner similar
            ↪ to what we do below. We then have:
            \begin{align*}
            |\sigma| \geq \max \Big\{\frac{d - 4n}{4\log_2 (n)} + 2, \frac{d}{2n}\Big\}.
            \end{align*}
            Since $\mu(\sigma) = \frac{d}{|\sigma|}$, the preceding bound immediately yields an upper bound on $\mu(\sigma)$:
            \begin{align*}
            \mu(\sigma) \leq \frac{d}{\max \Big\{\frac{d - 4n}{4\log_2(n)} + 4, \frac{d}{2n}\Big\}} = \min \Big\{\Big(\frac{d}{d - 4n + 8\log_2
            ↪ (n)}\Big)4\log_2 (n), 2n\Big\}.
            \end{align*}
            Now, we bound $\alpha(\sigma) = \frac{1}{d}\sum_{\sigma_i \in \sigma} |\sigma_i|^2$. Clearly, $\alpha(\sigma)$ grows with the size of
            ↪ the cycles in $\sigma$, so we may upper bound it by considering the largest size the cycles in $\sigma$ could be. Recall that
            ↪ $2\floor{2 \log_2(n)} \leq 2n$ for all $n \in \mathbb{N}$, so the cycles from Lemma \ref{lemma:bipartite_cycle_decomposition} are
            ↪ always smaller than those guaranteed by the extremal bound of $2n$. Thus, the more size-$2n$ cycles in our cycle decomposition, the
            ↪ higher $\alpha(\sigma)$. Since we can have at most $4n$ edges contribute to cycles of size at most $2n$, if $d \leq 4n$, then we have:
            \begin{align*}
            \alpha(\sigma) \leq \frac{1}{d} \frac{d}{2n} (2n)^2 = 2n.
            \end{align*}
            Otherwise, we have:
            \begin{align*}
            \alpha(\sigma) \leq \frac{1}{d} \Big(\frac{d - 4n}{2\floor{2 \log_2(n)}}(2\floor{2 \log_2(n)})^2 + \frac{4n}{2n}(2n)^2 \Big) \leq 4
            ↪ \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d},
            \end{align*}
            where the first inequality follows from computing the formula with all $4n$ edges in cycles at their maximum size of $2n$ and the
            ↪ remaining edges in cycles at their maximum size of $2\floor{2 \log_2(n)}$. The second inequality comes from rearranging terms and
            ↪ weakening $2\floor{2 \log_2(n)}$ to $4 \log_2(n)$. Putting the two cases together, we have:
            \begin{align*}
            \alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{d}, 2n\Big\}.
            \end{align*}
```

```
\end{proof}
```

[Next verbatim source excerpt]

```
\begin{proof}
    Applying Lemma \ref{lemma:Gc_cycle_decomposition} with $d = n2^{n-2}$ results in the following expressions:
    \begin{align*}
    \mu(\sigma) \leq \min \Big\{\Big(\frac{n2^{n-2}}{n2^{n-2} - 4n + 8\log_2 (n)}\Big)4\log_2 (n), 2n\Big\} \\
    \alpha(\sigma) \leq \min \Big\{4 \log_2(n) + \frac{4n(2n - 4 \log_2(n))}{n2^{n-2}}, 2n\Big\}.
    \end{align*}
    Clearly, as $n$ grows, the exponential growth of $d$ results in both bounds quickly converging to $4 \log_2(n)$. For small values of $n$, a
    ↪  simple plot immediately reveals that for $n \geq 2$, the expressions never exceed $5 \log_2(n)$.
\end{proof}
```

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

For every n >= 3, the all-even-subsets comparison frame has a cycle decomposition whose mean and dispersion are at most 5 log_2(n).

Archival source anchor: cited publication, lines 919-925

Expanded PaperInterface specification

```
∀ (n : ℕ) (hn : 3 ≤ n),
  ∃ (D :
    SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
      (SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame n
        (SeshadriUgander2020IIATesting.corollary_all_even_subsets_correctedSpec._proof_2 n hn))),
    D.cycleMean ≤ 5 * Real.logb 2 ↑n ∧
      SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion
        (SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame n
          (SeshadriUgander2020IIATesting.corollary_all_even_subsets_correctedSpec._proof_2 n hn)).incidenceCount
        D.length ≤
       5 * Real.logb 2 ↑n
```

Retained governing declarations

- SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame
- SeshadriUgander2020IIATesting.ChoiceFrame.incidenceCount
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean
- SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion

Lean-elaborated claim roles

```
1. parameter: ℕ
2. assumption: 3 ≤ n
3. conclusion: ∃ (D :
  SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
    (SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame n
      (SeshadriUgander2020IIATesting.corollary_all_even_subsets_correctedSpec._proof_2 n hn))),
  D.cycleMean ≤ 5 * Real.logb 2 ↑n ∧
    SeshadriUgander2020IIATesting.CycleMixture.cycleDispersion
      (SeshadriUgander2020IIATesting.AllEvenChoiceSet.frame n
        (SeshadriUgander2020IIATesting.corollary_all_even_subsets_correctedSpec._proof_2 n hn)).incidenceCount
      D.length ≤
     5 * Real.logb 2 ↑n
```

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.corollary_all_even_subsets_corrected

Recorded source-to-Spec screening

Verdict: matches_approved_corrected_target
Reason: The target uses the all-even-subsets comparison frame and states existence of a cycle decomposition whose mean and dispersion are both at most 5 log_2(n). Its n >= 3 range is the approved correction because the printed n = 2 endpoint is not Eulerian.
Reviewer check: ☐ Matches formalized target
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 11

Source locator: cited publication, lines 527-529

Verbatim source input

```
\begin{lemma}\label{lemma:seperation}
    A test between $p_0^N$ and ${\bar{q}_{\epsilon} }^N$, for any $N$ and $\epsilon \geq 2 \mu(\sigma) \delta$ is not statistically harder than the original IIA test. In particular, $ \epsilon \ge 4n \delta \geq 2\mu(\sigma)\delta$, suffices for all $\sigma$.
\end{lemma}
```

[Next verbatim source excerpt]

```
For the first simple step, consider the testing problem:
\begin{align}
\label{eq:test2}
\begin{cases}
H_0: (x, C) \sim p^N & \text{for } p = p_0 \\
H_1: (x,C) \sim q^N & \text{for some unknown } q \in \Mdelta,
\end{cases}
\end{align}
where $p_0$ is the uniform distribution, $p_{0,(x,C)} = 1/d, \forall (x,C)$ (recalling that each discete distribution is $d$-dimensional, where $d$ is the sum of the cardinalities of the choice sets in a given $\mathcal C$). Since $p_0 \in \mathcal{P}_0$, the problem is simpler than the original problem, and any lower bound on the performance of this problem carries over to the original problem of interest.

Now consider a discrete distribution $\qbe$ that is a perturbation of the uniform distribution of the form:
\begin{align*}
\qbe((x,C)) = \frac{1}{d} + \frac{\epsilon b_{(x,C)}}{d},
\end{align*}
\begin{align*}
\epsilon \in [0,1] \
b \in \lbrace -1, 1 \rbrace^d,
\
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\text{ and }
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x.
\end{align*}

That is, $b$ perturbs the uniform distribution such that there is no net translation over any set, or over all of the appearances of an item over sets. Our goal is to construct perturbations $\qbe$, selecting $\epsilon$ and $b$ that land in $\Mdelta$, meaning they are at least $\delta$ away (in TV distance) from $\Piia$ but at the same time still correspond to valid probability distributions (i.e., within the $d$-simplex). We are starting our journey (in the direction of $b$) at the uniform distribution $p_0$, and we want to show that we always land at least $\delta$ away from {\it any} distribution in $\Piia$.

Let $\Bset$ be any collection of $M=|\Bset|$ vectors $b$ satisfying the above conditions for the given collection $\mathcal C$. That is,
\begin{align}
\label{eq:bset_full}
\Bset \subseteq
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C \in \mathcal C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x
\right \}.
\end{align}
We define $\bar{q}_\epsilon = \frac{1}{M} \sum_{b \in \Bset} \qbe$ as the mixture distribution over the $M$ distributions in $\Bset$. Samples from $\bar{q}_\epsilon$ would be drawn marginally, that is, some $\qbe$ would be chosen uniformly from the $M$ possible distributions, and samples would then come from that $\qbe$. Consider then the testing problem:
```

[Next verbatim source excerpt]

```
\xhdr{Cycle decompositions and orientations of Eulerian bipartite graphs} To find a structured collection of Eulerian orientations of $G_\mathcal{C}$, we begin by examining the cycle decompositions of $G_\mathcal{C}$.
We first raise the simple fact that every Eulerian graph can be decomposed into edge-disjoint simple undirected cycles. We remind the reader that we call a decomposition of $G_\mathcal{C}$ into a set of $s$ simple cycles a {\it cycle decomposition}. There are ostensibly many different ways to select edge-disjoint cycles of $G_\mathcal{C}$, and we denote the collection of all possible edge-disjoint cycle decompositions of $G_\mathcal{C}$ by $\Sigma_{\mathcal{C}}$.

\begin{figure}[t]
    \centering
    \includegraphics[height=55mm]{iia_graph_illustration.pdf}
    \caption{
        A diagram illustrating the steps undertaken to construct Eulerian orientations for a simple example collection $\mathcal C$. (a) The hypergraph of the collection under study, $\mathcal{C} = \{\{a,b\},\{c,d\},\{a,b,c,d\}\}$. (b) The comparison incidence graph $G_{\mathcal C}$ corresponding to $\mathcal C$. (c) One possible cycle decomposition of $G_\mathcal{C}$, consisting of two undirected cycles. (d,e) Two different Eulerian orientations of the given cycle decomposition. In the first orientation, (d), both cycles are directed counterclockwise. In the second, (e), both cycles are directed clockwise. Mixtures of these two rotations produce two other orientations of the given cycle decomposition, not shown, yielding a total of four orientations.
    }
    \label{fig:gc}
    \vspace{-4mm}
\end{figure}
A cycle decomposition $\sigma \in \Sigma_\mathcal{C}$ is a set of cycles forming a feasible edge-disjoint cycle decomposition of $G_\mathcal{C}$, and an element $\sigma_i \in \sigma$ denotes the $i$th cycle in $\sigma$. We use $|\sigma|$ to denote the number of cycles in the decomposition $\sigma$, and $|\sigma_i|$ to denote the length of each cycle. Since all cycles of bipartite graphs have even length, we will write $|\sigma_i|=2k_i$, where $k_i$ is the number of item-nodes in the cycle.

Lastly, observe that we can always find a cycle decomposition of $G_\mathcal{C}$ such that every cycle has size at most $2n$. This observation follows from a simple application of the pigeonhole principle: since $G_\mathcal{C}$ is bipartite, with $n$ item-nodes and $m$ set-nodes, any cycle of length greater than $2n$ must visit some item-node more than once; hence the cycle is not simple, and can be decomposed into two smaller edge-disjoint cycles.
```

\xhdr{From cycle decomposition to perturbations}
We generate a structured collection of perturbation vectors b by generating many distinct Eulerian orientations of a given cycle decomposition σ. We can independently direct the edges in each of the cycles in σ in one of 2 directions (clockwise or counterclockwise), and all of these collections of directions correspond to valid b vectors. Thus, we have found $M=2^{|\sigma|}$ different Eulerian orientation for $G_\mathcal{C}$, each with a one-to-one correspondence to a valid perturbation vector b.

We define $\Bset(\sigma)$ as the {\it specific} collection of perturbation vectors b that orient a cycle decomposition σ, refining our generic definition of such collections of perturbations $\Bset$ in Equation~\eqref{eq:bset_full}:
\begin{align}
\label{eq:bset}
\Bset(\sigma) =
\left \{
b \in \lbrace -1, 1 \rbrace^d
:
\sum_{y \in C} b_{y,C} = 0, \forall C,
\ \
\sum_{\substack{C \in \mathcal{C}\\ C \ni x}} b_{x,C} = 0, \forall x,
\ \
|b_\ell - b_{\ell+1} | =2, \forall \ell \in \sigma_i,
\forall \sigma_i \in \sigma
\right \}.
\end{align}
Here (x,C) indexes the elements of the vector b and we use ℓ to index sequential edges in a cycle σ_i. For each perturbation vector $b \in \Bset(\sigma)$, we can construct a discrete distribution $\qbe$ as described previously. We additionally define $\mu(\sigma) = \frac{d}{|\sigma|}$, the average cycle length.

[Next verbatim source excerpt]

Since the entries associated with a simple cycle σ_i has a lower bound of $\frac{\epsilon}{2d}$, the lower bound on the objective is simply $\sum_{i \in [|\sigma|]} \frac{\epsilon}{2d} = \frac{\epsilon |\sigma|}{2d}$. The more cycles, the higher this lower bound is---thus, this quantity is at least $\frac{\epsilon}{2d} \frac{d}{2n} = \frac{\epsilon}{4n}$ even in the worst case. To summarize, we have shown that
\begin{align*}
\inf_{p \in \Piia} \tv{\qbe - p} \geq \frac{\epsilon |\sigma|}{2d} \geq \frac{\epsilon}{4n}
\end{align*}

Thus, if $\epsilon \geq \frac{2d}{|\sigma|}\delta = 2\mu(\sigma)\delta$, a sufficient condition for which is $\epsilon \geq 4n \delta$, then $\qbe \in \Mdelta$ for all $b \in \mathcal{B}_\mathcal{C}(\sigma)$.

Expanded PaperInterface specification

```
∀ {F : SeshadriUgander2020IIATesting.ChoiceFrame} (D : SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F)
  (W : D.AlternatingCycleWitness) (ε δ : ℝ) (hε_nonneg : 0 ≤ ε) (hε_le_one : ε ≤ 1),
  0 ≤ δ →
    ∀ (N : ℕ),
      (2 * D.cycleMean * δ ≤ ε →
         (∀ (a : D.Cycle → Bool),
            (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign a) ε hε_nonneg
                hε_le_one).SeparatedFromIIA
              δ) ∧
          ∀ (φ : SeshadriUgander2020IIATesting.FiniteDistribution.Test (Fin N → F.Observation)),
            ∃ (a : D.Cycle → Bool),
              ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N).binaryError
                  (SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts Finset.univ
                    (SeshadriUgander2020IIATesting.lemma2_testing_reductionSpec._proof_2 D)
                    (fun (orientation : D.Cycle → Bool) =>
                      (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign orientation) ε hε_nonneg
                          hε_le_one).asFiniteDistribution)
                    N)
                  φ ≤
                ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N).binaryError
                  ((SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign a) ε hε_nonneg
                        hε_le_one).asFiniteDistribution.product
                    N)
                  φ) ∧
      (D.cycleMean ≤ 2 * ↑(Fintype.card F.Item) → 4 * ↑(Fintype.card F.Item) * δ ≤ ε → 2 * D.cycleMean * δ ≤ ε)
```

Retained governing declarations

- SeshadriUgander2020IIATesting.ChoiceFrame
- SeshadriUgander2020IIATesting.ChoiceFrame.Observation
- SeshadriUgander2020IIATesting.ChoiceFrame.instDecidableEqObservation
- SeshadriUgander2020IIATesting.ChoiceFrame.instFintypeObservation
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.AlternatingCycleWitness
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.cycleMean
- SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition.orientedSign
- SeshadriUgander2020IIATesting.ChoiceSystem.SeparatedFromIIA
- SeshadriUgander2020IIATesting.ChoiceSystem.asFiniteDistribution
- SeshadriUgander2020IIATesting.ChoiceSystem.perturb

- SeshadriUgander2020IIATesting.ChoiceSystem.uniform
- SeshadriUgander2020IIATesting.FiniteDistribution.Test
- SeshadriUgander2020IIATesting.FiniteDistribution.binaryError
- SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts
- SeshadriUgander2020IIATesting.FiniteDistribution.product

Lean-elaborated claim roles

```
1. parameter: SeshadriUgander2020IIATesting.ChoiceFrame
2. parameter: SeshadriUgander2020IIATesting.ChoiceSystem.CycleDecomposition F
3. parameter: D.AlternatingCycleWitness
4. parameter: ℝ
5. parameter: ℝ
6. assumption: 0 ≤ ε
7. assumption: ε ≤ 1
8. assumption: 0 ≤ δ
9. parameter: ℕ
10. conclusion: (2 * D.cycleMean * δ ≤ ε →
   (∀ (a : D.Cycle → Bool),
      (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign a) ε hε_nonneg hε_le_one).SeparatedFromIIA
       δ) ∧
    ∀ (φ : SeshadriUgander2020IIATesting.FiniteDistribution.Test (Fin N → F.Observation)),
     ∃ (a : D.Cycle → Bool),
       ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N).binaryError
          (SeshadriUgander2020IIATesting.FiniteDistribution.mixtureOfProducts Finset.univ
            (SeshadriUgander2020IIATesting.lemma2_testing_reductionSpec._proof_2 D)
            (fun (orientation : D.Cycle → Bool) =>
              (SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign orientation) ε hε_nonneg
                 hε_le_one).asFiniteDistribution)
            N)
          φ ≤
        ((SeshadriUgander2020IIATesting.ChoiceSystem.uniform F).asFiniteDistribution.product N).binaryError
          ((SeshadriUgander2020IIATesting.ChoiceSystem.perturb (D.orientedSign a) ε hε_nonneg
                 hε_le_one).asFiniteDistribution.product
            N)
          φ) ∧
 (D.cycleMean ≤ 2 * ↑(Fintype.card F.Item) → 4 * ↑(Fintype.card F.Item) * δ ≤ ε → 2 * D.cycleMean * δ ≤ ε)
```

Lean proof endpoint (built separately)

SeshadriUgander2020IIATesting.lemma2_testing_reduction

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

Reason: The target retains both clauses of source Lemma 2 for every sample size: epsilon at least 2 mu(sigma) delta makes every oriented component delta-separated and makes the finite orientation-mixture experiment no harder than the original composite test, while epsilon at least 4n delta is sufficient when mu(sigma) is at most 2n. The alternating witness is the explicit Lean representation of the source's concrete alternating simple-cycle traversal; the mixture comparison uses averaging rather than the false nonconvex mixture-membership sentence in the printed proof, without changing the lemma's conclusion or constants.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation