Searchable Menus

Frank Yang ⓡ Piotr Dworczak*

Stanford University

Cologne, Northwestern & GRAPE

Cornell University
October 6, 2026

*Co-funded by the European Union (ERC, IMD-101040122). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council.

Multidimensional screening is hard...

Multidimensional screening is hard...

- No general (useful) **characterization** of optimal mechanisms.

Multidimensional screening is hard...

- No general (useful) **characterization** of optimal mechanisms.
- Optimal mechanism may be **complicated** even in simple instances of the problem (e.g., Manelli and Vincent, 2007).

Multidimensional screening is hard...

- No general (useful) **characterization** of optimal mechanisms.
- Optimal mechanism may be **complicated** even in simple instances of the problem (e.g., Manelli and Vincent, 2007).
- Optimal mechanisms may have **counterintuitive** properties (e.g., Hart and Reny, 2015).

Multidimensional screening is hard...

- No general (useful) **characterization** of optimal mechanisms.
- Optimal mechanism may be **complicated** even in simple instances of the problem (e.g., Manelli and Vincent, 2007).
- Optimal mechanisms may have **counterintuitive** properties (e.g., Hart and Reny, 2015).
- “Simple” mechanisms are generally **poor approximations** of fully optimal ones (e.g., Hart and Nisan, 2019).

Multidimensional screening is hard...

- No general (useful) **characterization** of optimal mechanisms.
- Optimal mechanism may be **complicated** even in simple instances of the problem (e.g., Manelli and Vincent, 2007).
- Optimal mechanisms may have **counterintuitive** properties (e.g., Hart and Reny, 2015).
- “Simple” mechanisms are generally **poor approximations** of fully optimal ones (e.g., Hart and Nisan, 2019).
- **Final nail in the coffin?** (Lahr and Niemeyer, 2025): The set of mechanisms uniquely optimal for some type distribution is **dense** in all IC and IR mechanisms—“anything goes.”

Multidimensional screening is important...

The approach of this paper

- **Observation:** In a complicated mechanism, the **agent** may have a hard time figuring out **what to do**.

The approach of this paper

- **Observation:** In a complicated mechanism, the **agent** may have a hard time figuring out **what to do**.
- We require that the mechanism makes the agent's problem of choosing the optimal option **simple**.

The approach of this paper

- **Observation:** In a complicated mechanism, the **agent** may have a hard time figuring out **what to do**.
- We require that the mechanism makes the agent's problem of choosing the optimal option **simple**.
- Specifically, the mechanism is an **ordered menu** where, for every type of the agent, **local maxima are global maxima**.

The approach of this paper

- **Observation:** In a complicated mechanism, the **agent** may have a hard time figuring out **what to do**.
- We require that the mechanism makes the agent's problem of choosing the optimal option **simple**.
- Specifically, the mechanism is an **ordered menu** where, for every type of the agent, **local maxima are global maxima**.
- Intuitively, **greedy search** leads to globally optimal behavior by the agent ($\Longrightarrow$ **searchable menu**).

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.
- We provide a **general characterization** of searchable menus.

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.
- We provide a **general characterization** of searchable menus.
- We study three **applications**:

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.
- We provide a **general characterization** of searchable menus.
- We study three **applications**:
    - multi-product monopolist;

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.
- We provide a **general characterization** of searchable menus.
- We study three **applications**:
  - multi-product monopolist;
  - screening with money and ordeals;

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.
- We provide a **general characterization** of searchable menus.
- We study three **applications**:
  - multi-product monopolist;
  - screening with money and ordeals;
  - income taxation with multidimensional heterogeneity.

Preview of results

- In **single-crossing** environments, the constraint has **no bite**.
- But in multidimensional settings, the constraint **restricts** the class of optimal mechanisms.
- We provide a **general characterization** of searchable menus.
- We study three **applications**:
  - multi-product monopolist;
  - screening with money and ordeals;
  - income taxation with multidimensional heterogeneity.
- **Extension**: a more permissive notion of **grid-searchability**.

Comments about the literature

- A huge literature in CS about imposing computational complexity constraints on the **mechanism designer**.

Comments about the literature

- A huge literature in CS about imposing computational complexity constraints on the **mechanism designer**.
- Equally large literature about the **performance of simple mechanisms** (vis à vis the optimal one).

Comments about the literature

- A huge literature in CS about imposing computational complexity constraints on the **mechanism designer**.
- Equally large literature about the **performance of simple mechanisms** (vis à vis the optimal one).
- Related work on **agent-side complexity/ tractability**:

  Jakobsen (2020), Camara (2025), Gonczarowski, Heffetz, and Thomas (2026).

Comments about the literature

- A huge literature in CS about imposing computational complexity constraints on the **mechanism designer**.
- Equally large literature about the **performance of simple mechanisms** (vis à vis the optimal one).
- Related work on **agent-side complexity/ tractability**:

  Jakobsen (2020), Camara (2025), Gonczarowski, Heffetz, and Thomas (2026).
- Sizable literature on **strategic simplicity**:

  Li (2017), Börgers and Li (2019), Mackenzie and Zhou (2022), Pycia and Troyan (2023), Li ⓡ Dworczak (2024), Nagel and Saitto (2026), Feffer and Tokarski (2026).

Comments about the literature

- A huge literature in CS about imposing computational complexity constraints on the **mechanism designer**.
- Equally large literature about the **performance of simple mechanisms** (vis à vis the optimal one).
- Related work on **agent-side complexity/ tractability**:

  Jakobsen (2020), Camara (2025), Gonczarowski, Heffetz, and Thomas (2026).
- Sizable literature on **strategic simplicity**:

  Li (2017), Börgers and Li (2019), Mackenzie and Zhou (2022), Pycia and Troyan (2023), Li ⓡ Dworczak (2024), Nagel and Saitto (2026), Feffer and Tokarski (2026).
- **Searchability and unrestricted optimality** need not conflict:

  Haghpanah and Hartline (2021), Bergemann, Bonatti, Haupt and Smolin (2021), Ghili (2023), Yang (2025, 2026).

Model

Setup

- **Outcome space** X and **type space** Θ.

Setup

- **Outcome space** X and **type space** Θ.

- Unit mass of agents indexed by types $\theta \in \Theta$, distributed according to $\mu \in \Delta(\Theta)$.

Setup

- **Outcome space** X and **type space** Θ.
- Unit mass of agents indexed by types $\theta \in \Theta$, distributed according to $\mu \in \Delta(\Theta)$.
- Each type θ has utility $u(x, \theta)$ over outcomes $x \in X$.

Setup

- **Outcome space** X and **type space** Θ.
- Unit mass of agents indexed by types $\theta \in \Theta$, distributed according to $\mu \in \Delta(\Theta)$.
- Each type θ has utility $u(x, \theta)$ over outcomes $x \in X$.
- **Outside option** $\underline{x} \in X$ with $u(\underline{x}, \theta) = 0$ for all θ.

Mechanisms

- A **(direct-revelation) mechanism** is a map $x : \Theta \to X$.

Mechanisms

- A **(direct-revelation) mechanism** is a map $x : \Theta \to X$.
- **IC** and **IR** defined as usual.

Mechanisms

- A **(direct-revelation) mechanism** is a map $x : \Theta \to X$.
- **IC** and **IR** defined as usual.
- Mechanism x is **implemented** by a menu $M \subseteq X$ (always assumed to contain the outside option) if, for each θ,

$$x(\theta) \in \operatorname{argmax}_{m \in M} \{ u(m, \theta) \}.$$

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).
- Assume that we can index options by $s \in S \subseteq \mathbb{R}$ with $x^s \preceq x^t \iff s \leq t$, and write $M = \{x^s\}_{s \in S}$.

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).
- Assume that we can index options by $s \in S \subseteq \mathbb{R}$ with $x^s \preceq x^t \iff s \leq t$, and write $M = \{x^s\}_{s \in S}$.
- **Technical assumptions:**

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).
- Assume that we can index options by $s \in S \subseteq \mathbb{R}$ with $x^s \preceq x^t \iff s \leq t$, and write $M = \{x^s\}_{s \in S}$.
- **Technical assumptions:**
  - the index set S is compact;

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).
- Assume that we can index options by $s \in S \subseteq \mathbb{R}$ with $x^s \preceq x^t \iff s \leq t$, and write $M = \{x^s\}_{s \in S}$.
- **Technical assumptions:**
  - the index set S is compact;
  - $u(x^s, \theta)$ is upper-semicontinuous in s for every θ.

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).
- Assume that we can index options by $s \in S \subseteq \mathbb{R}$ with $x^s \preceq x^t \iff s \leq t$, and write $M = \{x^s\}_{s \in S}$.
- **Technical assumptions:**
  - the index set S is compact;
  - $u(x^s, \theta)$ is upper-semicontinuous in s for every θ.
- Each type has a **well-defined optimal choice** in any menu.

Menus

- An **(ordered) menu** $(M, \preceq)$ is a totally ordered set of outcomes (containing the outside option).
- Assume that we can index options by $s \in S \subseteq \mathbb{R}$ with $x^s \preceq x^t \iff s \leq t$, and write $M = \{x^s\}_{s \in S}$.
- **Technical assumptions:**
  - the index set S is compact;
  - $u(x^s, \theta)$ is upper-semicontinuous in s for every θ.
- Each type has a **well-defined optimal choice** in any menu.
- (Feel free to think about finite menus.)

Searchable menu

- Interval $[\underline{s}, \overline{s}] \subset S$ is a **neighborhood** of $s \in S$ if either

$$\underline{s} < s < \overline{s}, \quad \underline{s} < s = \overline{s} = \max S, \quad \text{or} \quad \min S = \underline{s} = s < \overline{s}.$$

- A point $s \in S$ is a **local maximum** of $g : S \to \mathbb{R}$ if s maximizes g on some neighborhood of s.

Searchable menu

- Interval $[\underline{s}, \overline{s}] \subset S$ is a **neighborhood** of $s \in S$ if either

$$\underline{s} < s < \overline{s}, \quad \underline{s} < s = \overline{s} = \max S, \quad \text{or} \quad \min S = \underline{s} = s < \overline{s}.$$

- A point $s \in S$ is a **local maximum** of $g : S \to \mathbb{R}$ if s maximizes g on some neighborhood of s.

Definition

A menu $(M, \preceq)$ is **searchable** if, for every type θ:

- Every **local maximum** of $u(x^s, \theta)$ is a **global maximum**;
- Regularity: the set of global maximizers is an interval.

Searchable menu

- Interval $[\underline{s}, \overline{s}] \subset S$ is a **neighborhood** of $s \in S$ if either

$$\underline{s} < s < \overline{s}, \quad \underline{s} < s = \overline{s} = \max S, \quad \text{or} \quad \min S = \underline{s} = s < \overline{s}.$$

- A point $s \in S$ is a **local maximum** of $g : S \to \mathbb{R}$ if s maximizes g on some neighborhood of s.

Definition

A menu $(M, \preceq)$ is **searchable** if, for every type θ:

- Every **local maximum** of $u(x^s, \theta)$ is a **global maximum**;
- Regularity: the set of global maximizers is an interval.

Intuition: "greedy search" finds the agent's optimal choice (can be formalized by one-step dominance of Pycia and Troyan, 2023). Details

No bite in one-dimensional environments

- Outcome space $X \subseteq \mathbb{R}^2$ with $(q, p) \in X$, and type space $\Theta \subseteq \mathbb{R}$;
- $u((q, p), \theta)$ is strictly decreasing in p;
- **Single-crossing:** for $q < q'$, $\theta < \theta'$, $(q, p), (q', p') \in X$,

$$u((q, p), \theta) \leq u((q', p'), \theta) \implies u((q, p), \theta') < u((q', p'), \theta').$$

No bite in one-dimensional environments

- Outcome space $X \subseteq \mathbb{R}^2$ with $(q, p) \in X$, and type space $\Theta \subseteq \mathbb{R}$;
- $u((q, p), \theta)$ is strictly decreasing in p;
- **Single-crossing:** for $q < q'$, $\theta < \theta'$, $(q, p), (q', p') \in X$,

$$u((q, p), \theta) \leq u((q', p'), \theta) \implies u((q, p), \theta') < u((q', p'), \theta').$$

Proposition

In single-crossing environments, a mechanism is ***incentive-compatible*** *if and only if it is* ***searchable****.*

No bite in one-dimensional environments

- Outcome space $X \subseteq \mathbb{R}^2$ with $(q, p) \in X$, and type space $\Theta \subseteq \mathbb{R}$;
- $u((q, p), \theta)$ is strictly decreasing in p;
- **Single-crossing:** for $q < q'$, $\theta < \theta'$, $(q, p), (q', p') \in X$,

$$u((q, p), \theta) \leq u((q', p'), \theta) \implies u((q, p), \theta') < u((q', p'), \theta').$$

Proposition

In single-crossing environments, a mechanism is ***incentive-compatible*** *if and only if it is* ***searchable****.*

Sketch of proof:

- IC $+$ single-crossing $\Rightarrow$ **monotone allocation**

No bite in one-dimensional environments

- Outcome space $X \subseteq \mathbb{R}^2$ with $(q, p) \in X$, and type space $\Theta \subseteq \mathbb{R}$;
- $u((q, p), \theta)$ is strictly decreasing in p;
- **Single-crossing:** for $q < q'$, $\theta < \theta'$, $(q, p), (q', p') \in X$,

$$u((q, p), \theta) \leq u((q', p'), \theta) \implies u((q, p), \theta') < u((q', p'), \theta').$$

Proposition

In single-crossing environments, a mechanism is ***incentive-compatible*** *if and only if it is* ***searchable****.*

Sketch of proof:

- IC $+$ single-crossing $\Rightarrow$ **monotone allocation**
- Choose the order $\preceq$ of increasing allocations.

No bite in one-dimensional environments

- Outcome space $X \subseteq \mathbb{R}^2$ with $(q, p) \in X$, and type space $\Theta \subseteq \mathbb{R}$;
- $u((q, p), \theta)$ is strictly decreasing in p;
- **Single-crossing:** for $q < q'$, $\theta < \theta'$, $(q, p), (q', p') \in X$,

$$u((q, p), \theta) \leq u((q', p'), \theta) \implies u((q, p), \theta') < u((q', p'), \theta').$$

Proposition

In single-crossing environments, a mechanism is ***incentive-compatible*** *if and only if it is* ***searchable****.*

Sketch of proof:

- IC $+$ single-crossing $\Rightarrow$ **monotone allocation**
- Choose the order $\preceq$ of increasing allocations.
- Then, each type's utility is **semi-strictly quasi-concave** on the menu (strictly increasing, flat, then strictly decreasing). $\square$

Results

Incremental nesting

- For a **finite menu**, index by $S = \{1, \ldots, m\}$.

Incremental nesting

- For a **finite menu**, index by $S = \{1, \ldots, m\}$.
- **Weak incremental better-off sets:** $H^1 := \Theta$; for $k \in \{2, \ldots, m\}$,

$$H^k := \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$$

Incremental nesting

- For a **finite menu**, index by $S = \{1, \ldots, m\}$.
- **Weak incremental better-off sets:** $H^1 := \Theta$; for $k \in \{2, \ldots, m\}$,

$$H^k := \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$$

- **Strict incremental better-off sets:** $G^1 = \Theta$; for $k \in \{2, \ldots, m\}$,

$$G^k := \{\theta : u(x^{k-1}, \theta) < u(x^k, \theta)\}$$

Incremental nesting

- For a **finite menu**, index by $S = \{1, \ldots, m\}$.
- **Weak incremental better-off sets:** $H^1 := \Theta$; for $k \in \{2, \ldots, m\}$,

$$H^k := \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$$

- **Strict incremental better-off sets:** $G^1 = \Theta$; for $k \in \{2, \ldots, m\}$,

$$G^k := \{\theta : u(x^{k-1}, \theta) < u(x^k, \theta)\}$$

- **Incremental nesting condition** (INC):

$$H^{k+1} \subseteq H^k \text{ and } G^{k+1} \subseteq G^k, \forall k.$$

Main characterization

Theorem

A menu $(M, \preceq)$ *is* ***searchable*** *if and only if every finite submenu of* $(M, \preceq)$ *satisfies the* ***incremental nesting condition****.*

Main characterization

Theorem

A menu $(M, \preceq)$ *is* ***searchable*** *if and only if every finite submenu of* $(M, \preceq)$ *satisfies the* ***incremental nesting condition****.*

If M *is* ***finite****, it suffices to check the condition for* M *itself.*

Main characterization

Theorem

A menu $(M, \preceq)$ *is* ***searchable*** *if and only if every finite submenu of* $(M, \preceq)$ *satisfies the* ***incremental nesting condition****.*

If M *is* ***finite****, it suffices to check the condition for* M *itself.*

Comments:

- Searchability is a **type-by-type** condition.

Main characterization

Theorem

A menu $(M, \preceq)$ *is* ***searchable*** *if and only if every finite submenu of* $(M, \preceq)$ *satisfies the* ***incremental nesting condition****.*

If M *is* ***finite****, it suffices to check the condition for* M *itself.*

Comments:

- Searchability is a **type-by-type** condition.
- INC is an **across-type** condition.

Main characterization

Theorem

A menu $(M, \preceq)$ *is* ***searchable*** *if and only if every finite submenu of* $(M, \preceq)$ *satisfies the* ***incremental nesting condition****.*

If M *is* ***finite****, it suffices to check the condition for* M *itself.*

Comments:

- Searchability is a **type-by-type** condition.
- INC is an **across-type** condition.
- **Proof** is by direct application of the definitions (using equivalence of searchability and semi-strict quasi-concavity of agent's utilities over the menu).

Main characterization

Theorem

*A menu $(M, \preceq)$ is **searchable** if and only if every finite submenu of $(M, \preceq)$ satisfies the **incremental nesting condition**.*

*If M is **finite**, it suffices to check the condition for M itself.*

Comments:

- Searchability is a **type-by-type** condition.
- INC is an **across-type** condition.
- **Proof** is by direct application of the definitions (using equivalence of searchability and semi-strict quasi-concavity of agent's utilities over the menu).
- Under regularity, it suffices to check the weak INC.

Application 1: Multi-product monopoly

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:
  - $X = [0, 1]^n \times \mathbb{R}, \quad \Theta = \mathbb{R}^n_+$;

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:
  - $X = [0, 1]^n \times \mathbb{R}, \quad \Theta = \mathbb{R}^n_+$;
  - utility $u(x, \theta) = q \cdot \theta - p$, with $q \in [0, 1]^n$, $p \in \mathbb{R}$.

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:
  - $X = [0, 1]^n \times \mathbb{R}, \quad \Theta = \mathbb{R}^n_+$;
  - utility $u(x, \theta) = q \cdot \theta - p$, with $q \in [0, 1]^n$, $p \in \mathbb{R}$.

- μ has **full support**, continuous density, finite first moments.

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:
  - $X = [0, 1]^n \times \mathbb{R}, \quad \Theta = \mathbb{R}^n_+$;
  - utility $u(x, \theta) = q \cdot \theta - p$, with $q \in [0, 1]^n$, $p \in \mathbb{R}$.
- μ has **full support**, continuous density, finite first moments.
- Designer maximizes **expected revenue**.

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:
  - $X = [0, 1]^n \times \mathbb{R}, \quad \Theta = \mathbb{R}^n_+$;
  - utility $u(x, \theta) = q \cdot \theta - p$, with $q \in [0, 1]^n$, $p \in \mathbb{R}$.
- μ has **full support**, continuous density, finite first moments.
- Designer maximizes **expected revenue**.
- $(M, \preceq)$ is an **upgrade menu** if $q^s \leq q^t$ for all $s < t$.

Application 1: Multi-product monopoly

- n goods, quasi-linear preferences, additive values:
  - $X = [0, 1]^n \times \mathbb{R}, \quad \Theta = \mathbb{R}^n_+$;
  - utility $u(x, \theta) = q \cdot \theta - p$, with $q \in [0, 1]^n$, $p \in \mathbb{R}$.
- μ has **full support**, continuous density, finite first moments.
- Designer maximizes **expected revenue**.
- $(M, \preceq)$ is an **upgrade menu** if $q^s \leq q^t$ for all $s < t$.
- Number of **tiers** $= |M| - 1$.

Application 1: Multi-product monopoly

Theorem

There exists an optimal searchable menu $(M, \preceq)$ *such that* M *is an* ***upgrade menu with at most*** n ***tiers****, and the highest tier is the* ***grand bundle****.*

Application 1: Multi-product monopoly

Theorem

There exists an optimal searchable menu $(M, \preceq)$ *such that* M *is an* ***upgrade menu with at most*** n ***tiers****, and the highest tier is the* ***grand bundle****.*

Comments:

- Sharp contrast to “anything goes” results without searchability.

Application 1: Multi-product monopoly

Theorem

There exists an optimal searchable menu $(M, \preceq)$ *such that* M *is an* ***upgrade menu with at most*** n ***tiers****, and the highest tier is the* ***grand bundle****.*

Comments:

- Sharp contrast to "anything goes" results without searchability.
- Menu-size complexity $\leq$ **number of goods**, for any continuous distribution.

Application 1: Multi-product monopoly

Theorem

There exists an optimal searchable menu $(M, \preceq)$ *such that* M *is an* ***upgrade menu with at most*** n ***tiers****, and the highest tier is the* ***grand bundle****.*

Comments:

- Sharp contrast to "anything goes" results without searchability.
- Menu-size complexity $\leq$ **number of goods**, for any continuous distribution.
- FOSD shifts of values increase the monopolist's profit under searchability (cf. Hart and Reny, 2015).

Application 1: Multi-product monopoly

Theorem

There exists an optimal searchable menu $(M, \preceq)$ *such that* M *is an* ***upgrade menu with at most*** n ***tiers****, and the highest tier is the* ***grand bundle****.*

Comments:

- Sharp contrast to "anything goes" results without searchability.
- Menu-size complexity $\leq$ **number of goods**, for any continuous distribution.
- FOSD shifts of values increase the monopolist's profit under searchability (cf. Hart and Reny, 2015).
- Upgrade menus are **undominated** (Manelli and Vincent, 2007).

Multi-product monopoly **(proof sketch)**

- In a payoff type space, searchability $\iff$ WINC:

$$H^{k+1} \subseteq H^k, \forall k,$$

where $H^k = \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$ (and $H^1 = \Theta$).

Multi-product monopoly **(proof sketch)**

- In a payoff type space, searchability $\iff$ WINC:

$$H^{k+1} \subseteq H^k, \forall k,$$

  where $H^k = \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$ (and $H^1 = \Theta$).

- Sets H^k pin down optimal choices:

Multi-product monopoly **(proof sketch)**

- In a payoff type space, searchability $\iff$ WINC:

$$H^{k+1} \subseteq H^k, \forall k,$$

  where $H^k = \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$ (and $H^1 = \Theta$).

- Sets H^k pin down optimal choices:
  - $\theta \in H^m$ chooses option m

Multi-product monopoly **(proof sketch)**

- In a payoff type space, searchability $\iff$ WINC:

$$H^{k+1} \subseteq H^k, \forall k,$$

  where $H^k = \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$ (and $H^1 = \Theta$).

- Sets H^k pin down optimal choices:
  - $\theta \in H^m$ chooses option m
  - $\theta \in H^{m-1} \setminus H^m$ chooses option $m - 1$

Multi-product monopoly **(proof sketch)**

- In a payoff type space, searchability $\iff$ WINC:

$$H^{k+1} \subseteq H^k, \forall k,$$

  where $H^k = \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$ (and $H^1 = \Theta$).

- Sets H^k pin down optimal choices:
  - $\theta \in H^m$ chooses option m
  - $\theta \in H^{m-1} \setminus H^m$ chooses option $m - 1$
  - ...

Multi-product monopoly **(proof sketch)**

- In a payoff type space, searchability $\iff$ WINC:

$$H^{k+1} \subseteq H^k, \forall k,$$

  where $H^k = \{\theta : u(x^{k-1}, \theta) \leq u(x^k, \theta)\}$ (and $H^1 = \Theta$).

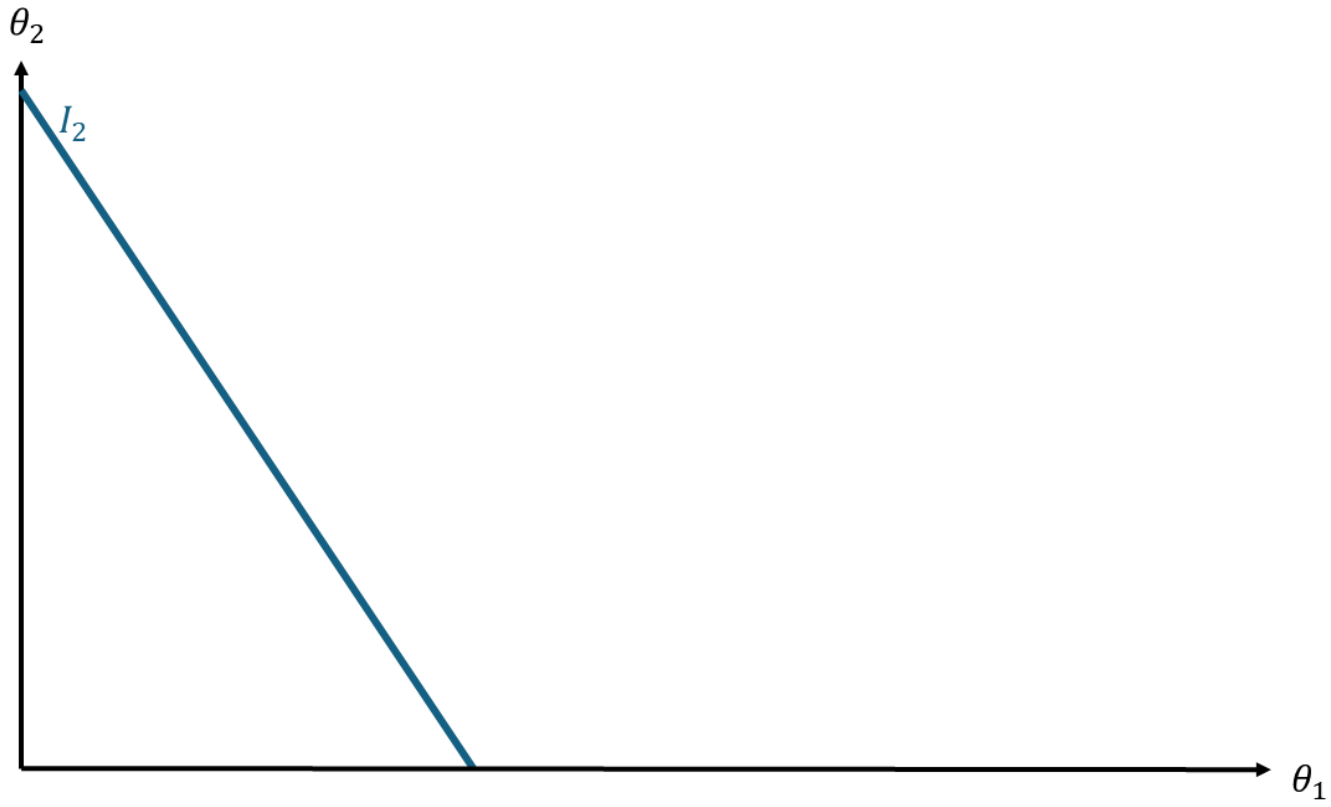
- Sets H^k pin down optimal choices:
  - $\theta \in H^m$ chooses option m
  - $\theta \in H^{m-1} \setminus H^m$ chooses option $m - 1$
  - ...

- Define indifference regions:

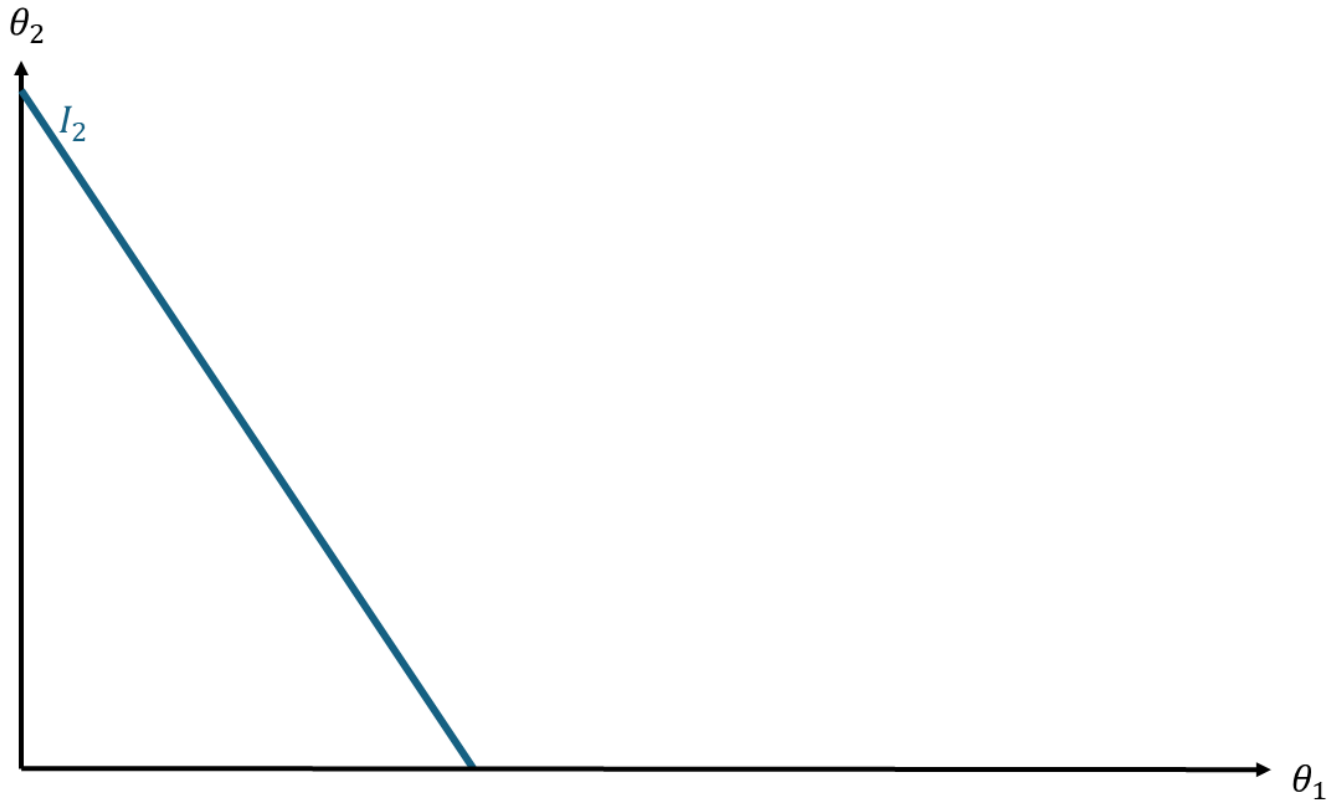
$$I^k := \{\theta : u(x^{k-1}, \theta) = u(x^k, \theta)\}.$$

Multi-product monopoly **(proof sketch)**

Multi-product monopoly **(proof sketch)**



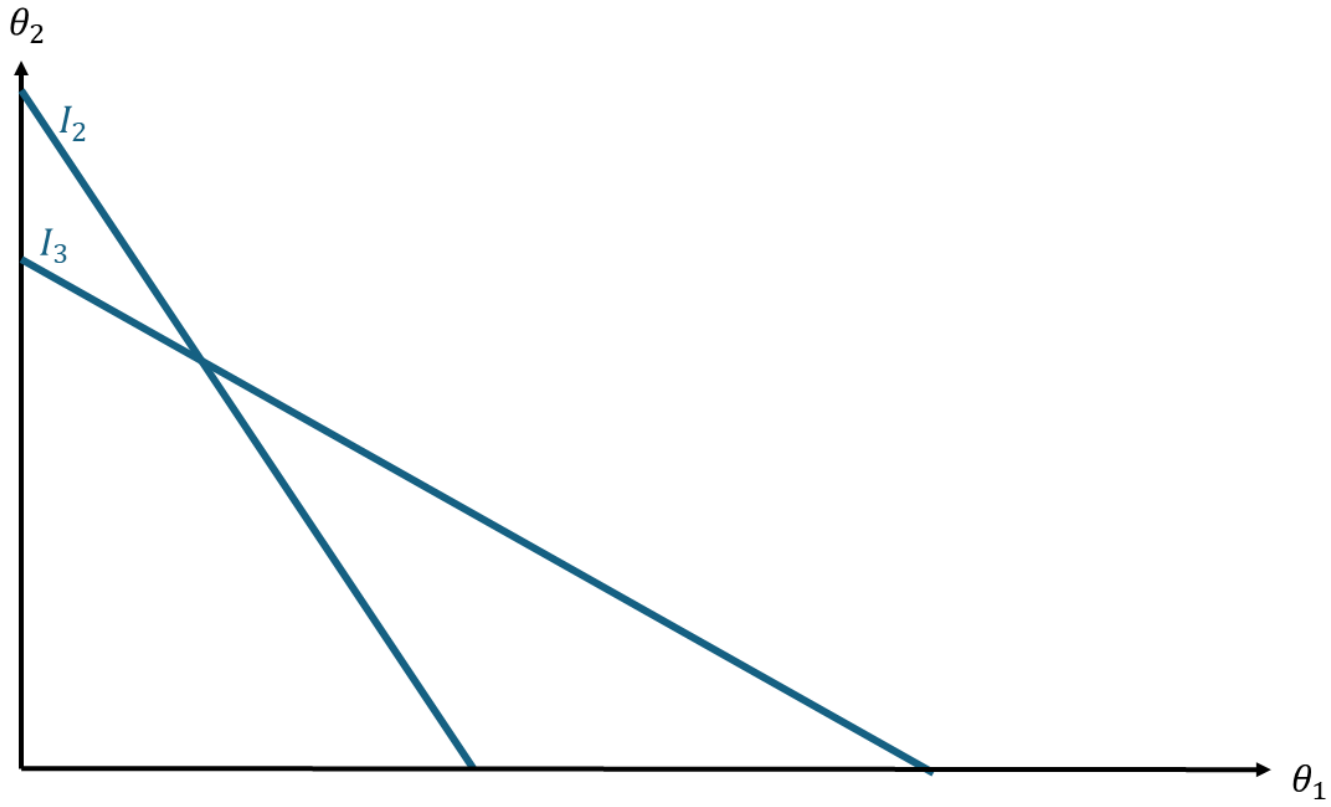
Multi-product monopoly **(proof sketch)**



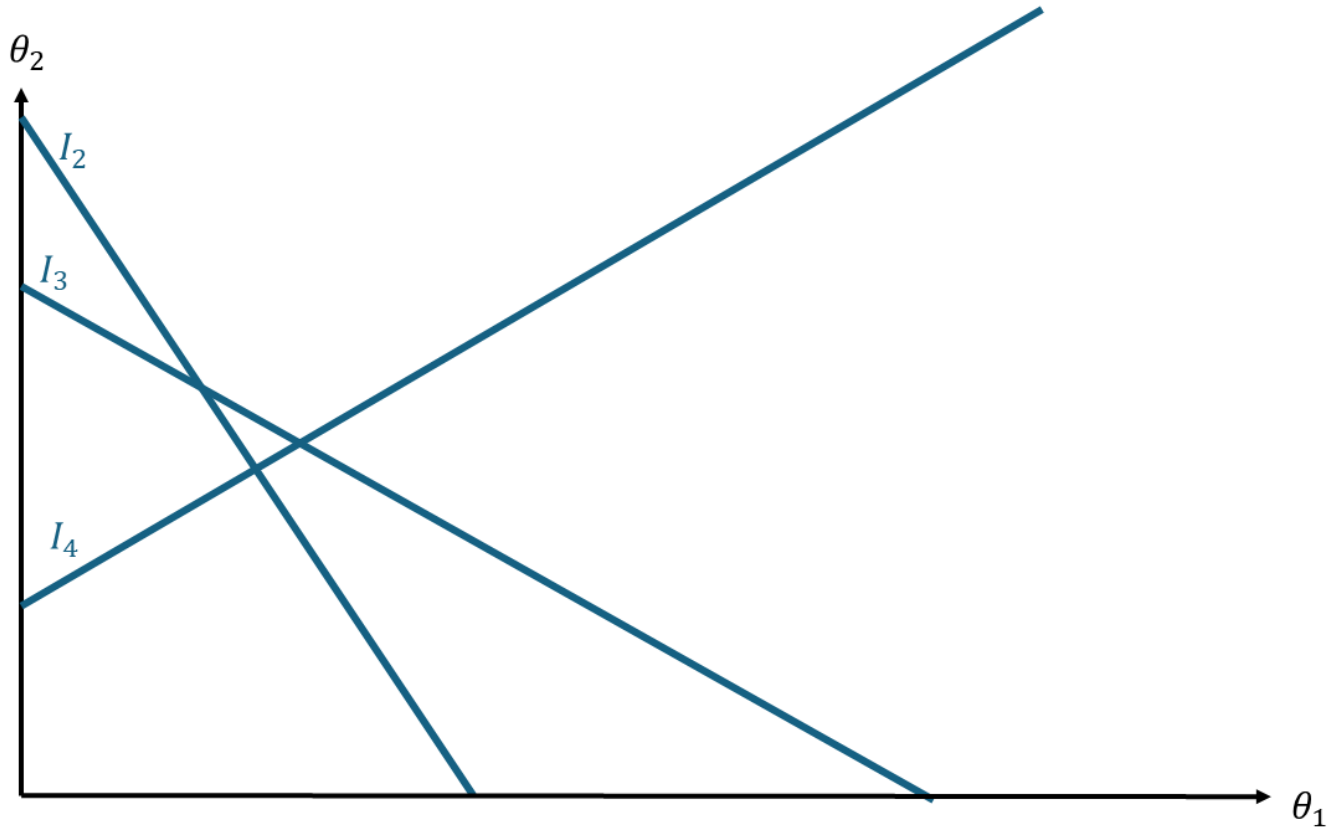
Let $\Delta q^k = q^k - q^{k-1}$ and $\Delta p^k = p^k - p^{k-1}$.

$$I^k = \{\theta : \theta \cdot \Delta q^k = \Delta p^k\}$$

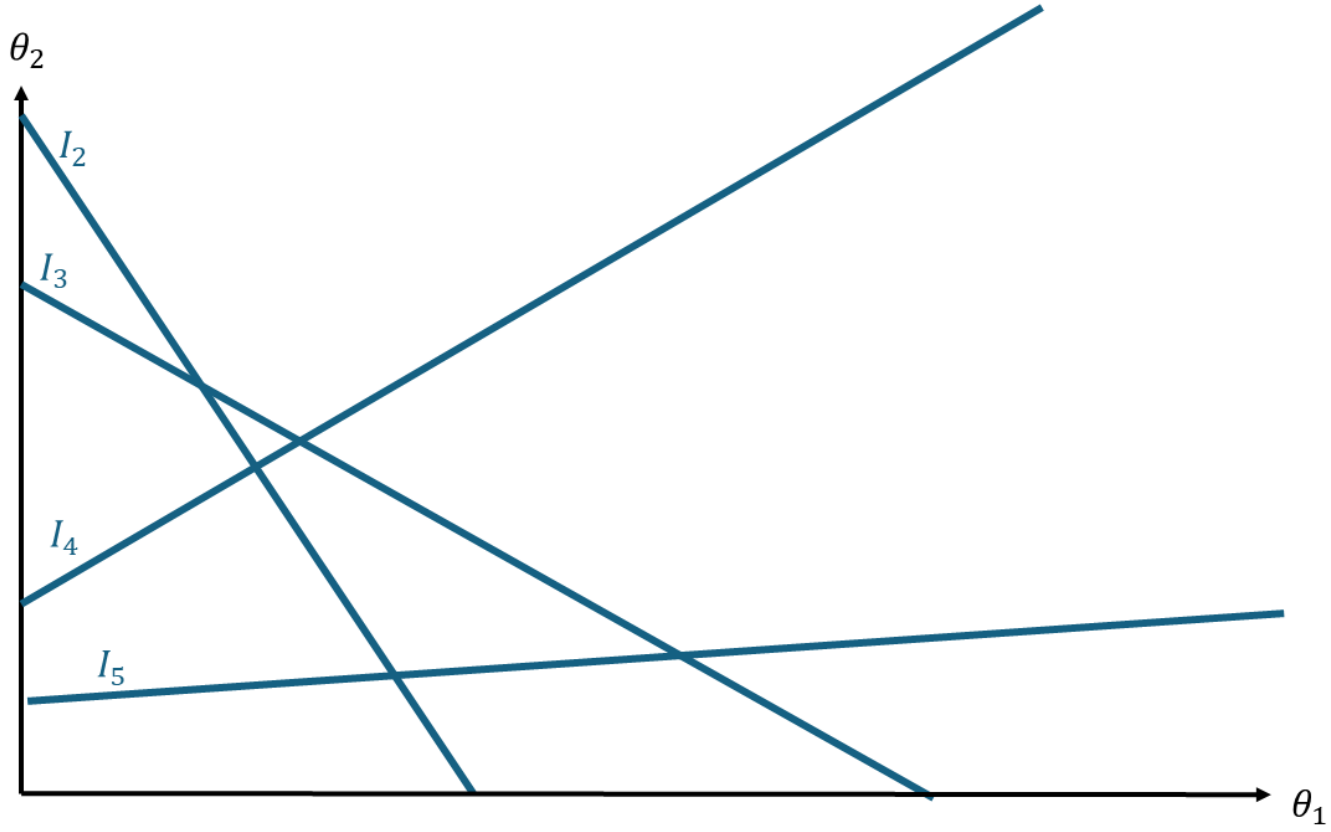
Multi-product monopoly **(proof sketch)**



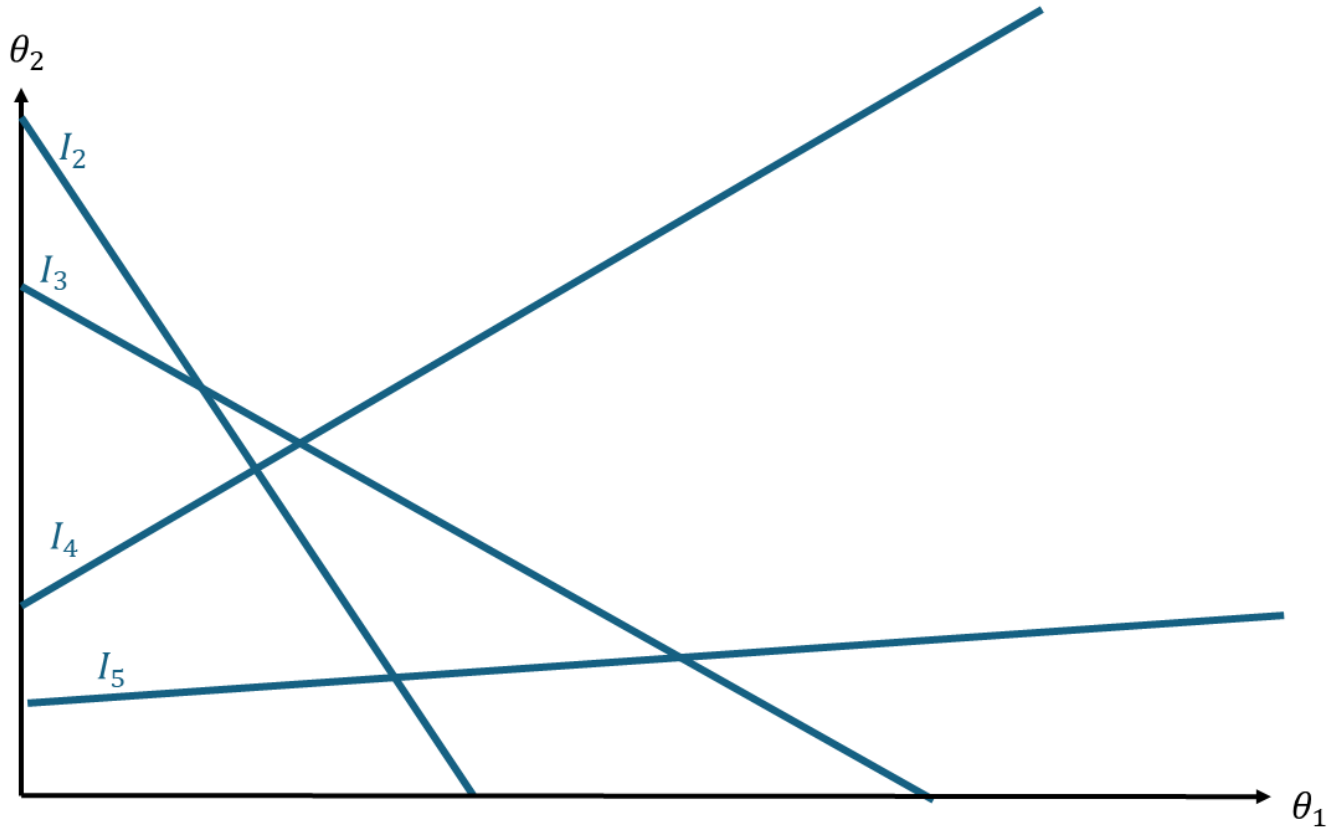
Multi-product monopoly **(proof sketch)**



Multi-product monopoly **(proof sketch)**

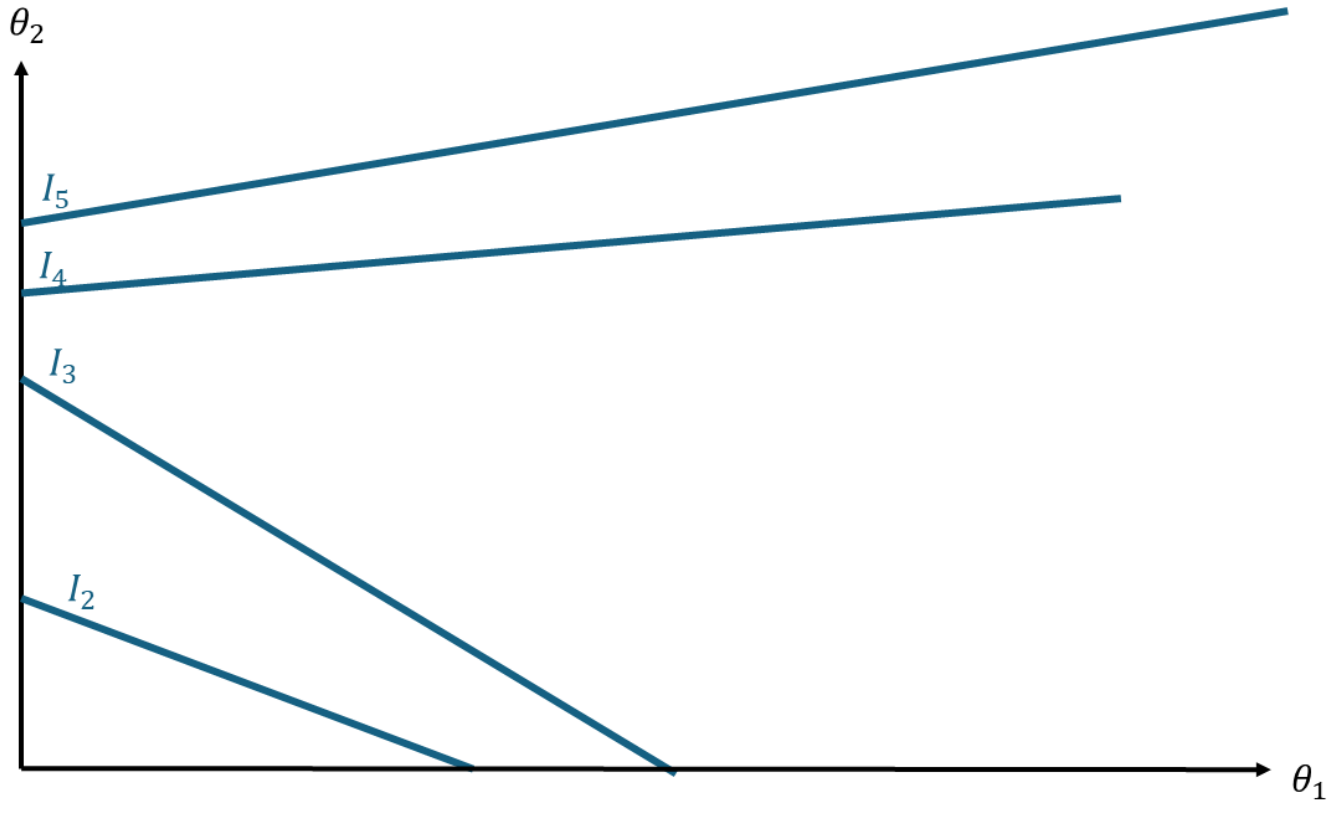


Multi-product monopoly **(proof sketch)**



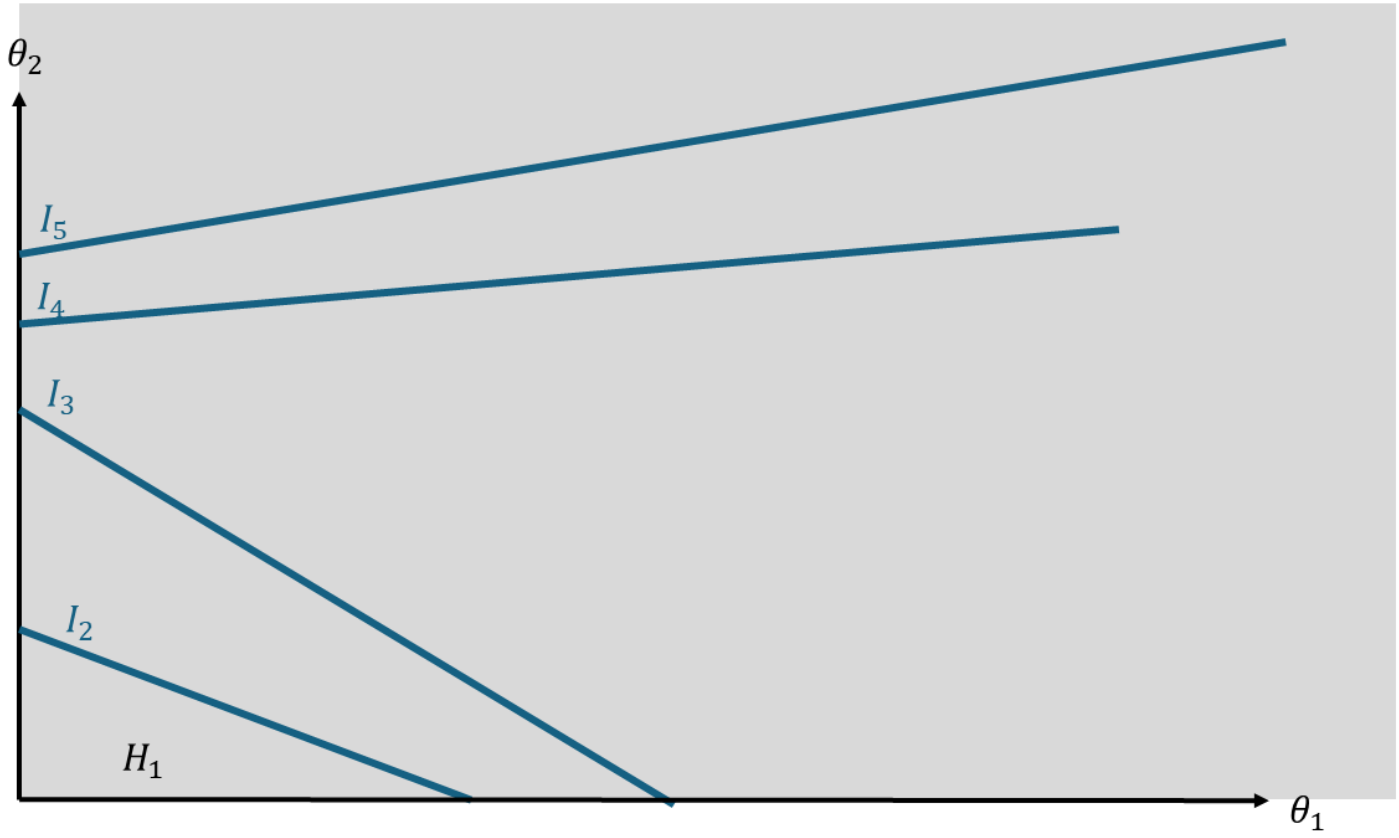
Incremental nesting implies no crossing of these hyperplanes.

Multi-product monopoly **(proof sketch)**

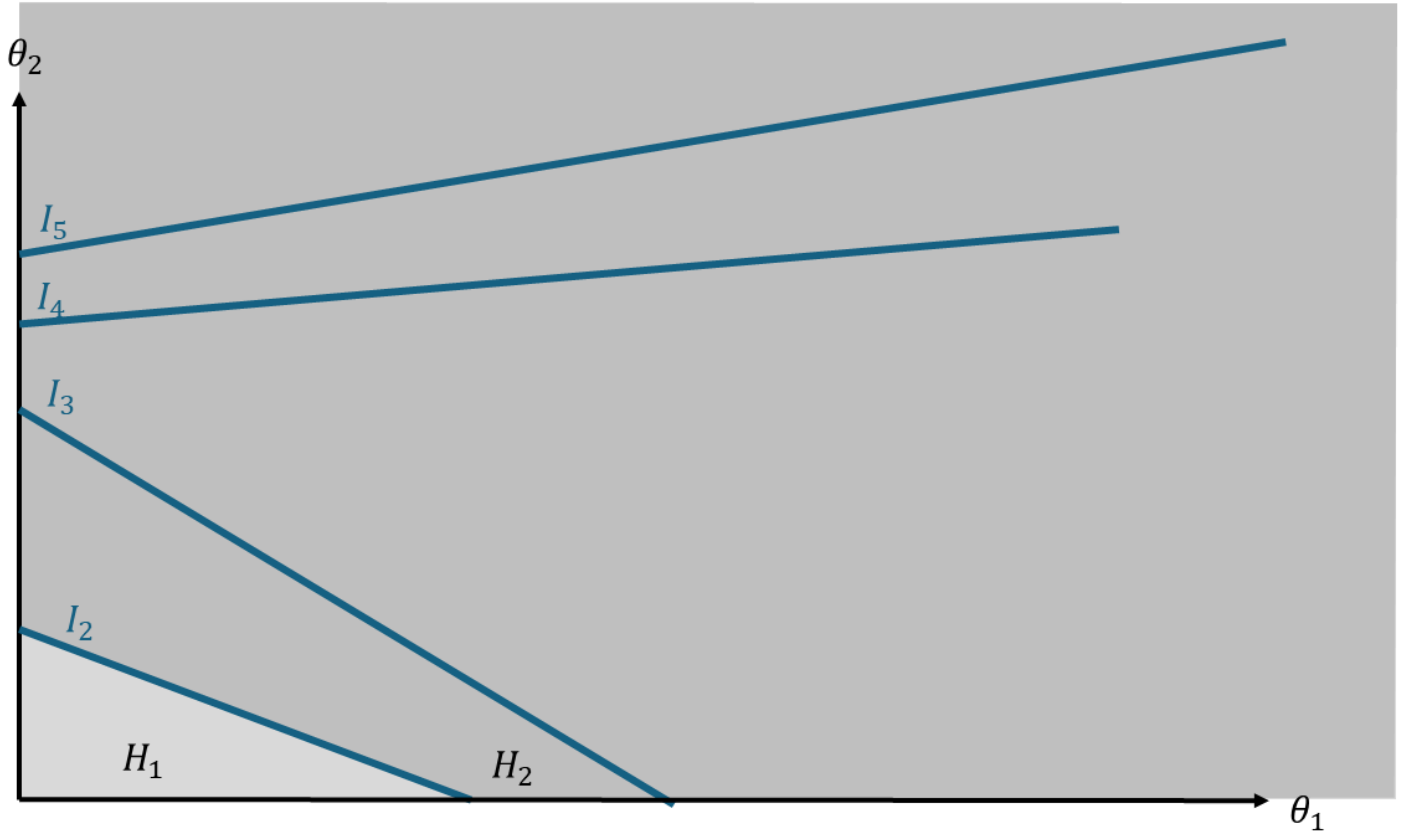


Incremental nesting implies no crossing of these hyperplanes.

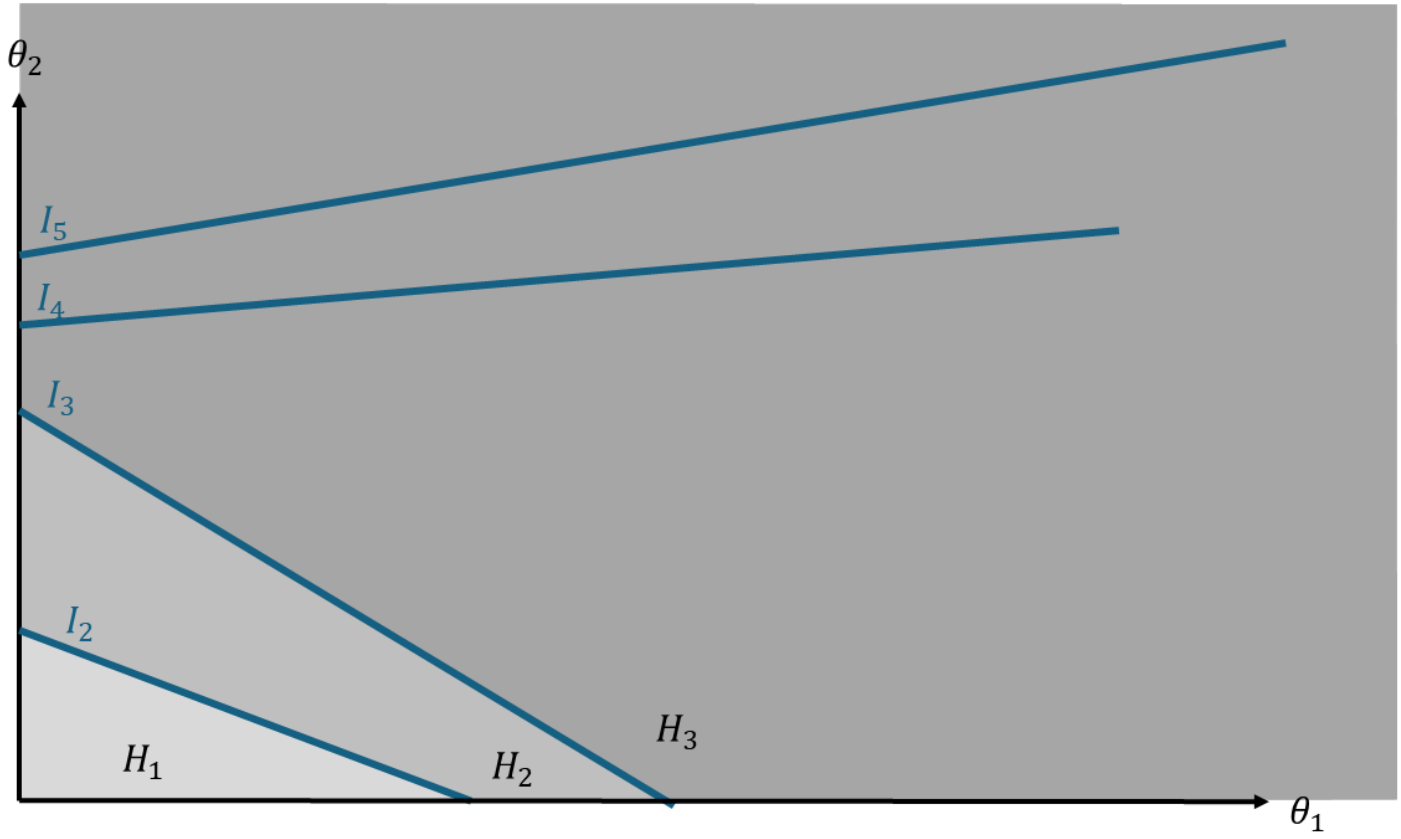
Multi-product monopoly **(proof sketch)**



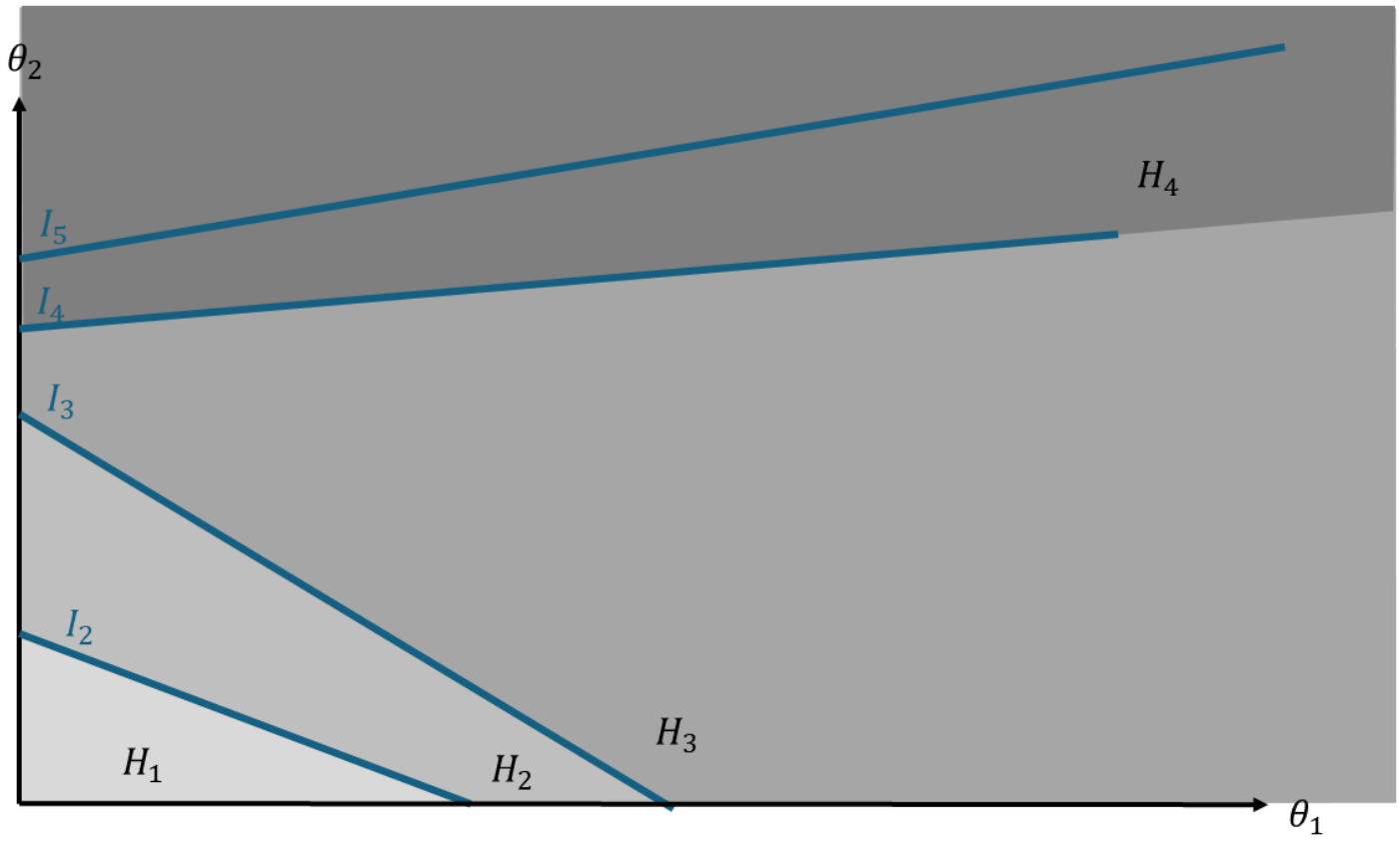
Multi-product monopoly **(proof sketch)**



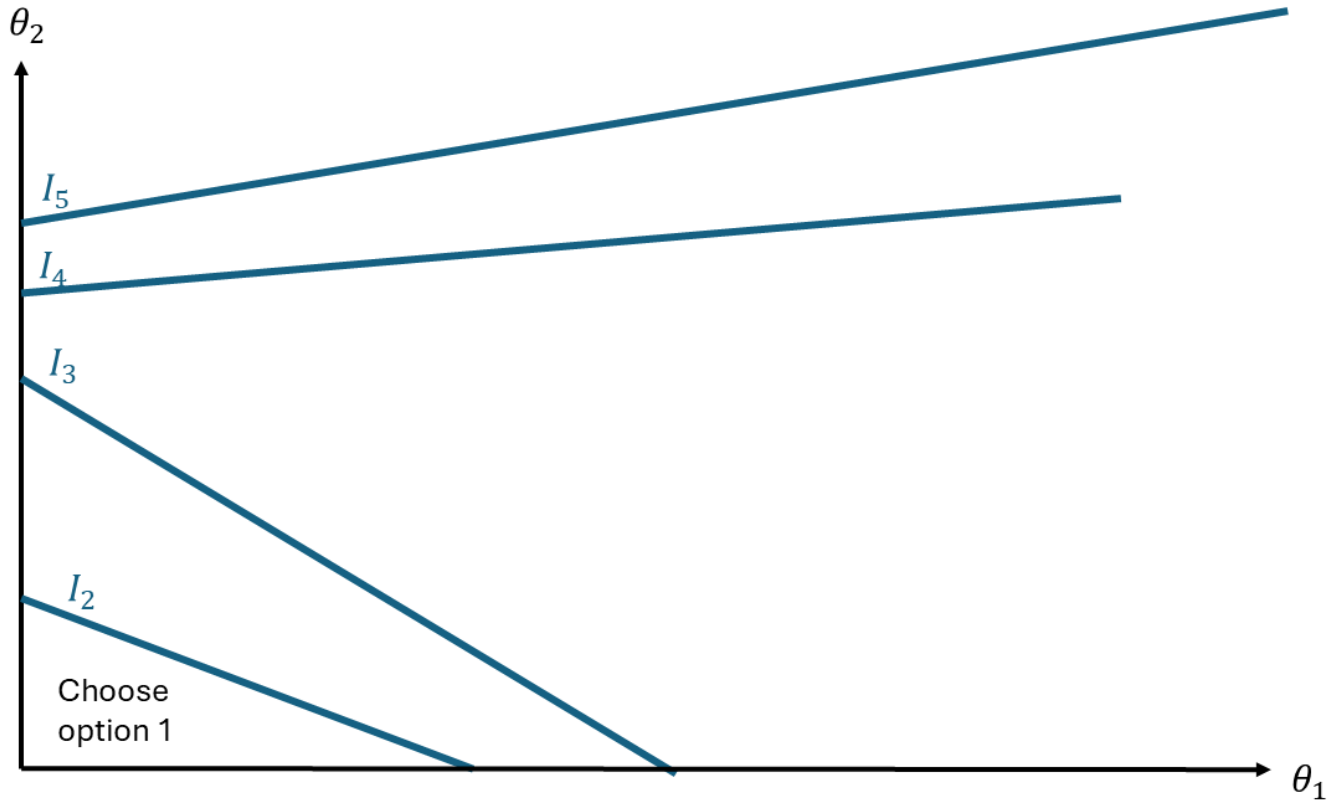
Multi-product monopoly **(proof sketch)**



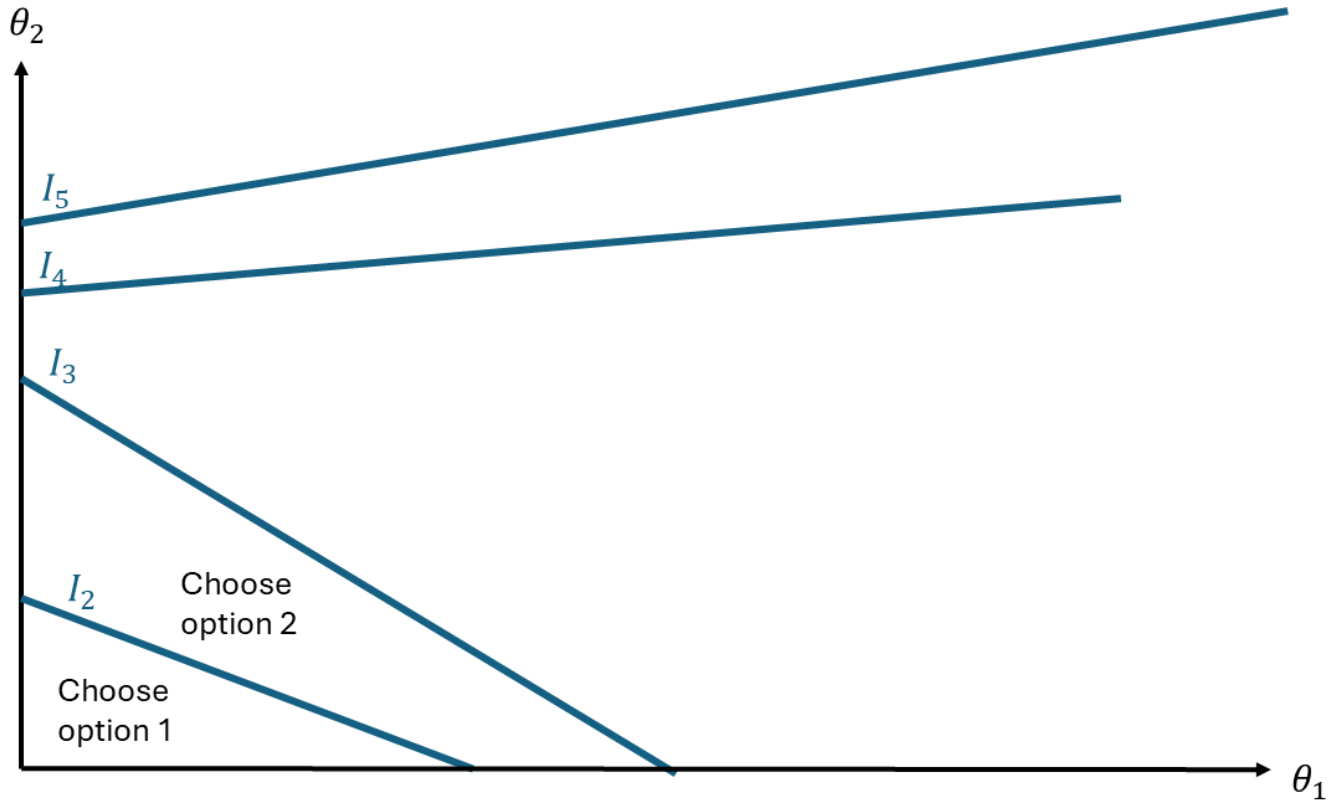
Multi-product monopoly **(proof sketch)**



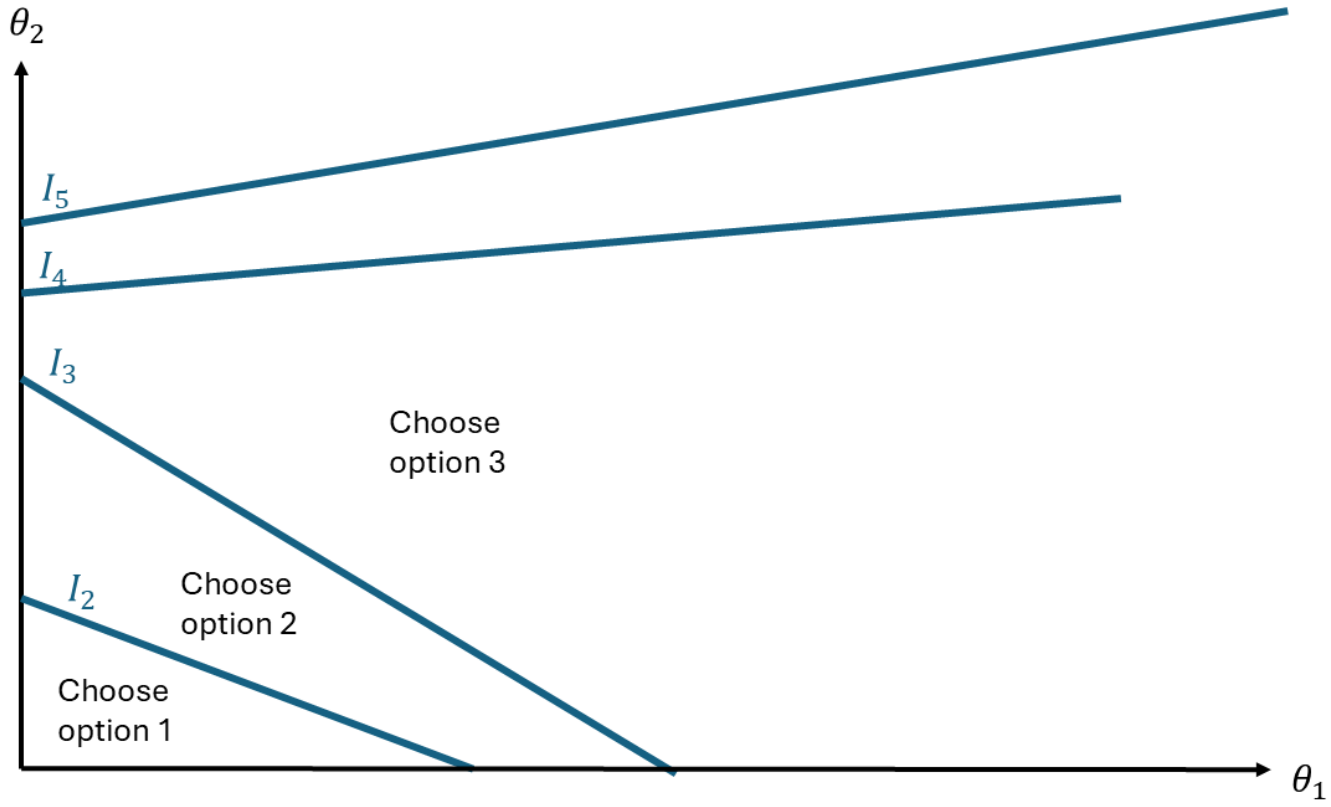
Multi-product monopoly **(proof sketch)**



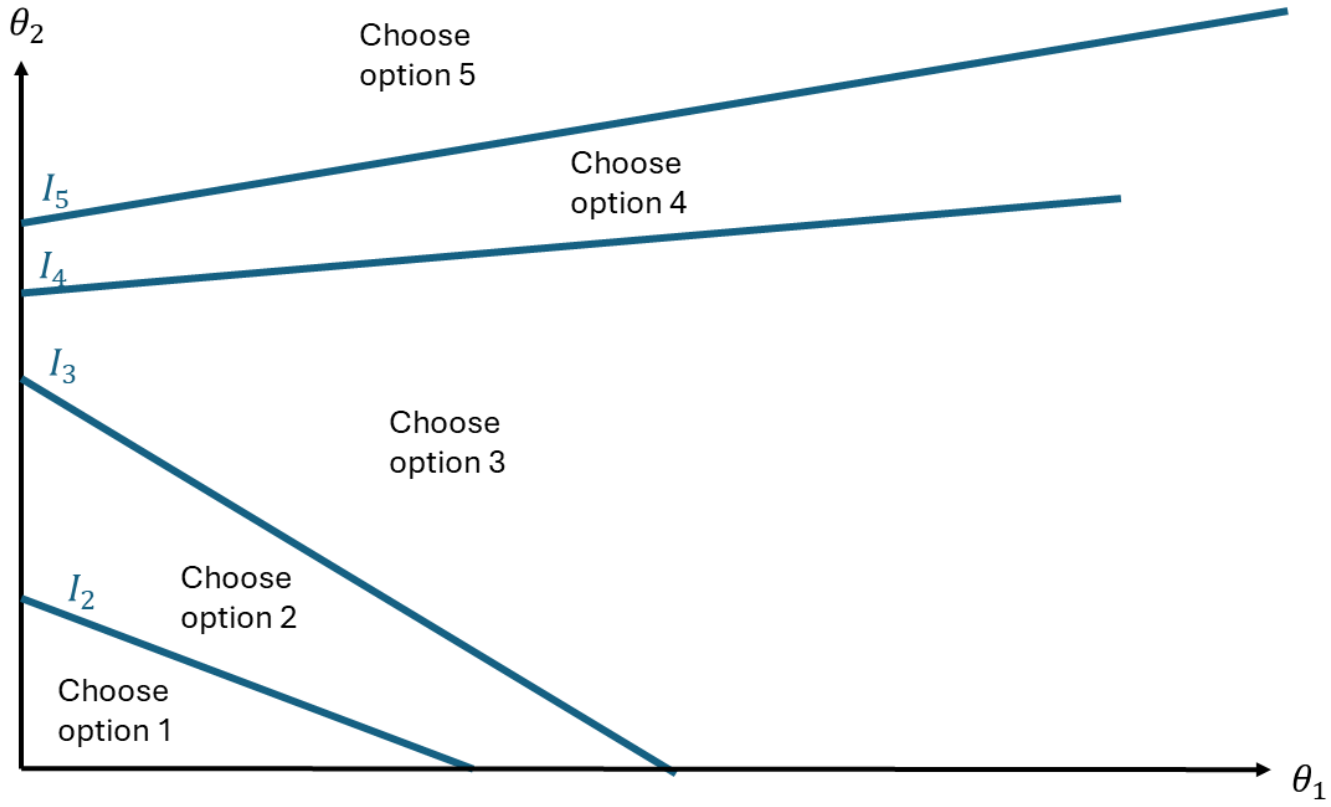
Multi-product monopoly **(proof sketch)**



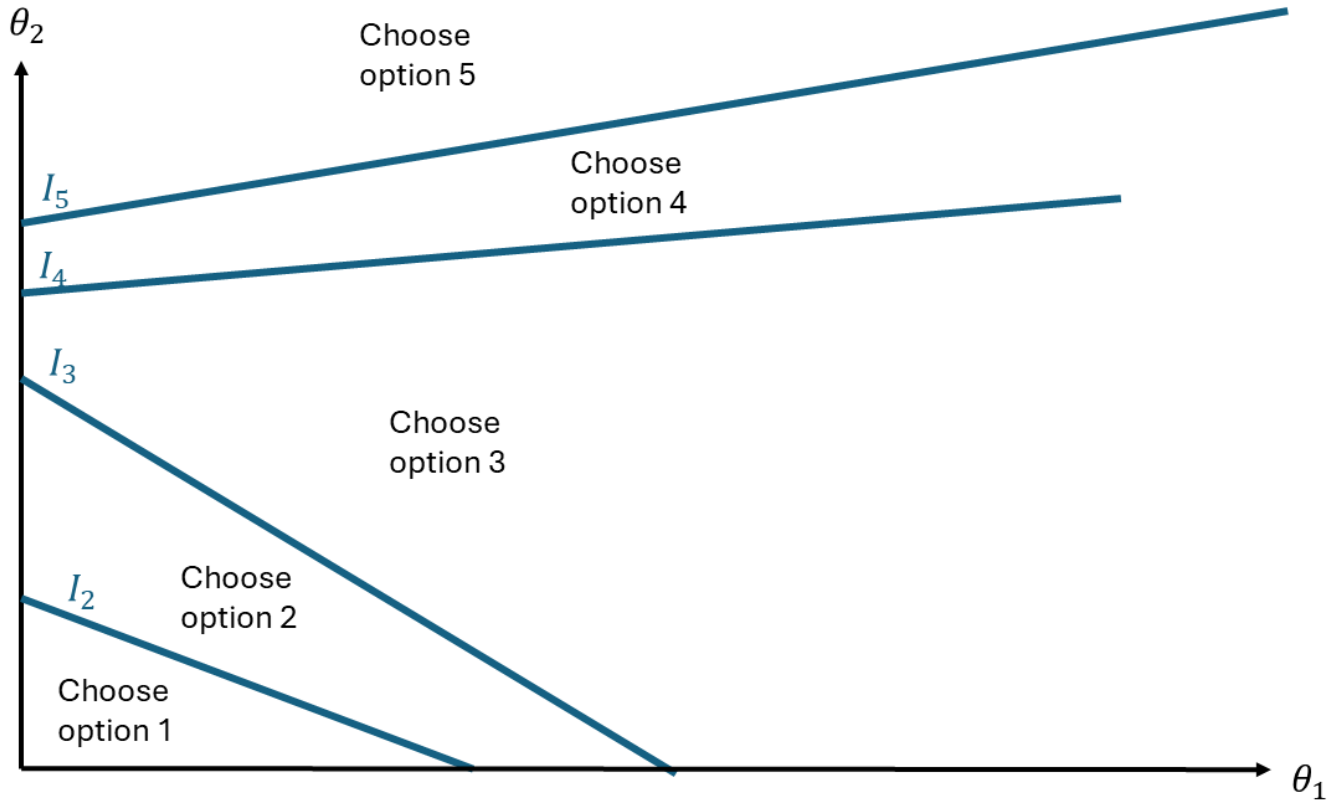
Multi-product monopoly **(proof sketch)**



Multi-product monopoly **(proof sketch)**

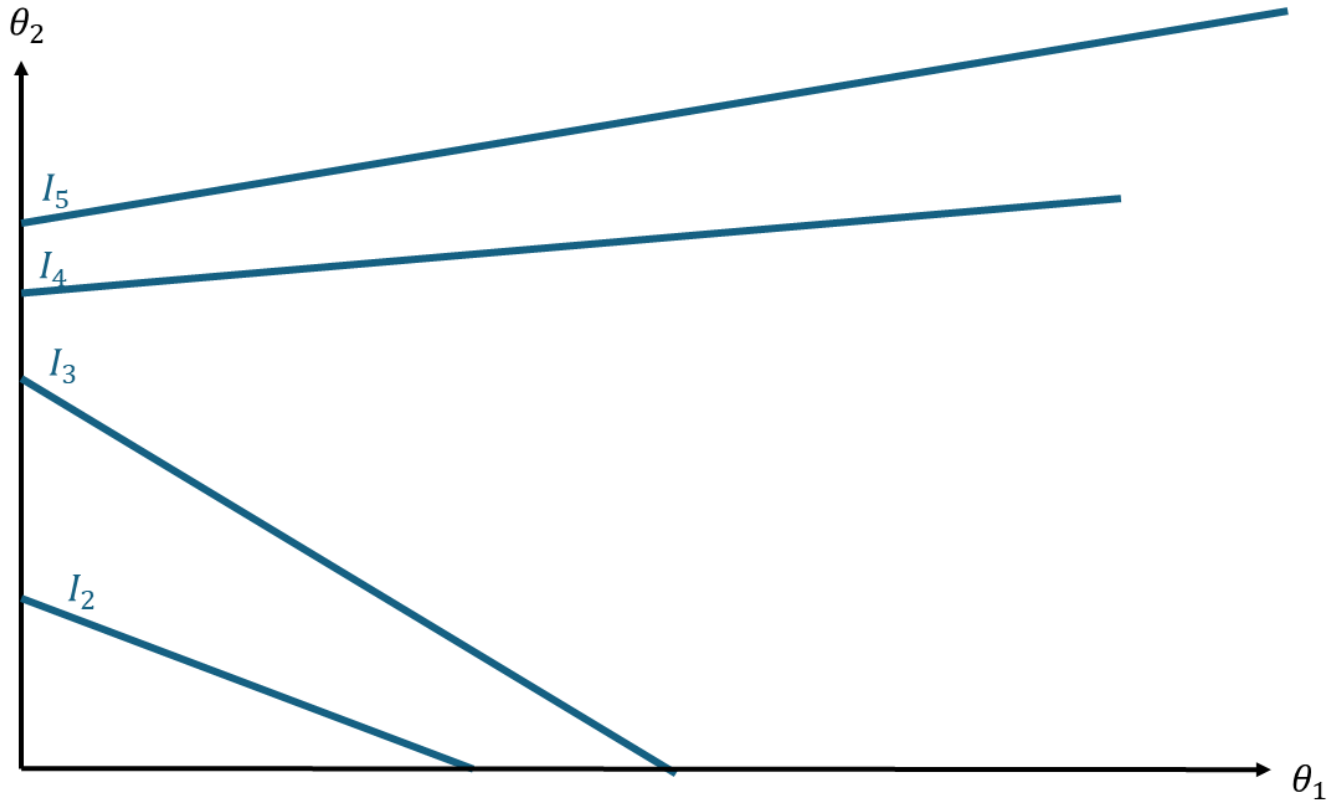


Multi-product monopoly **(proof sketch)**

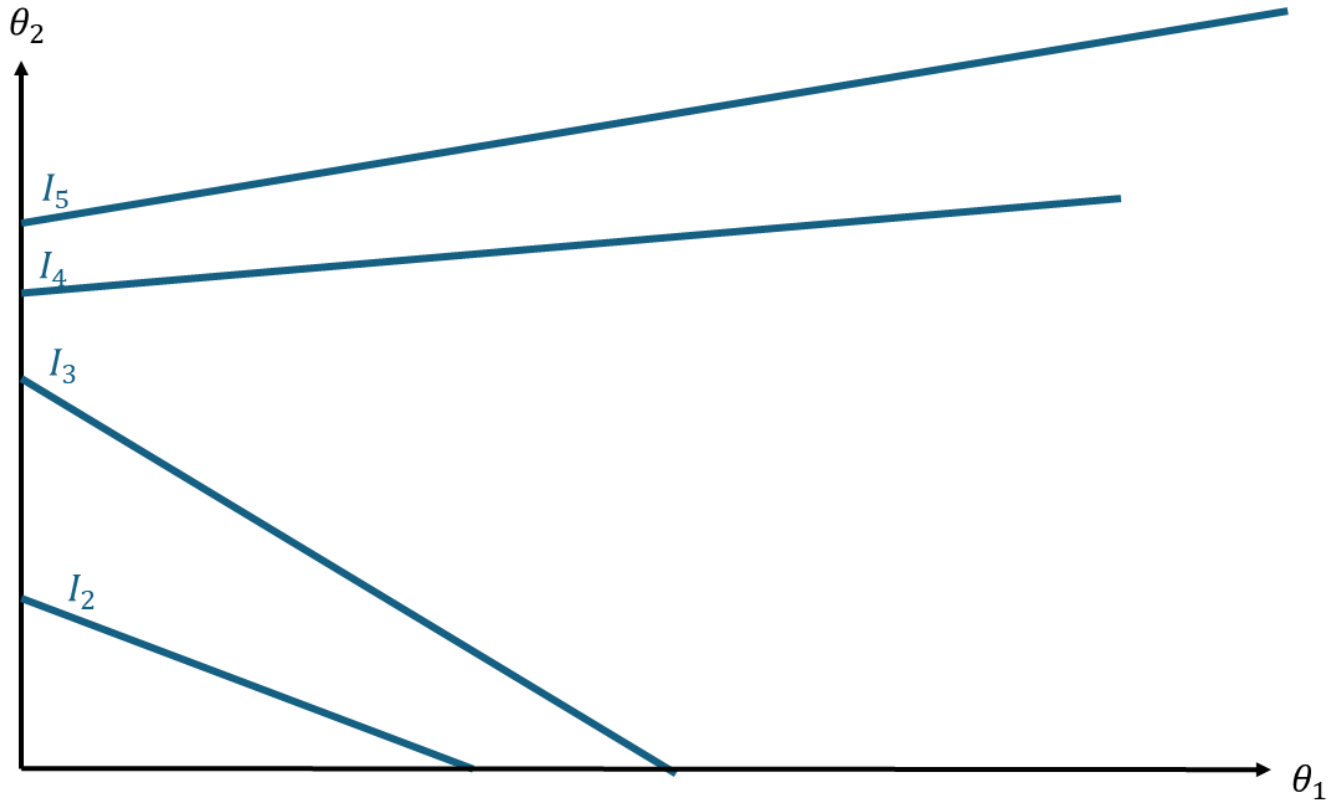


$$Revenue = \sum_{k=1}^{m} \Delta p^k \mu(H^k).$$

Multi-product monopoly **(proof sketch)**

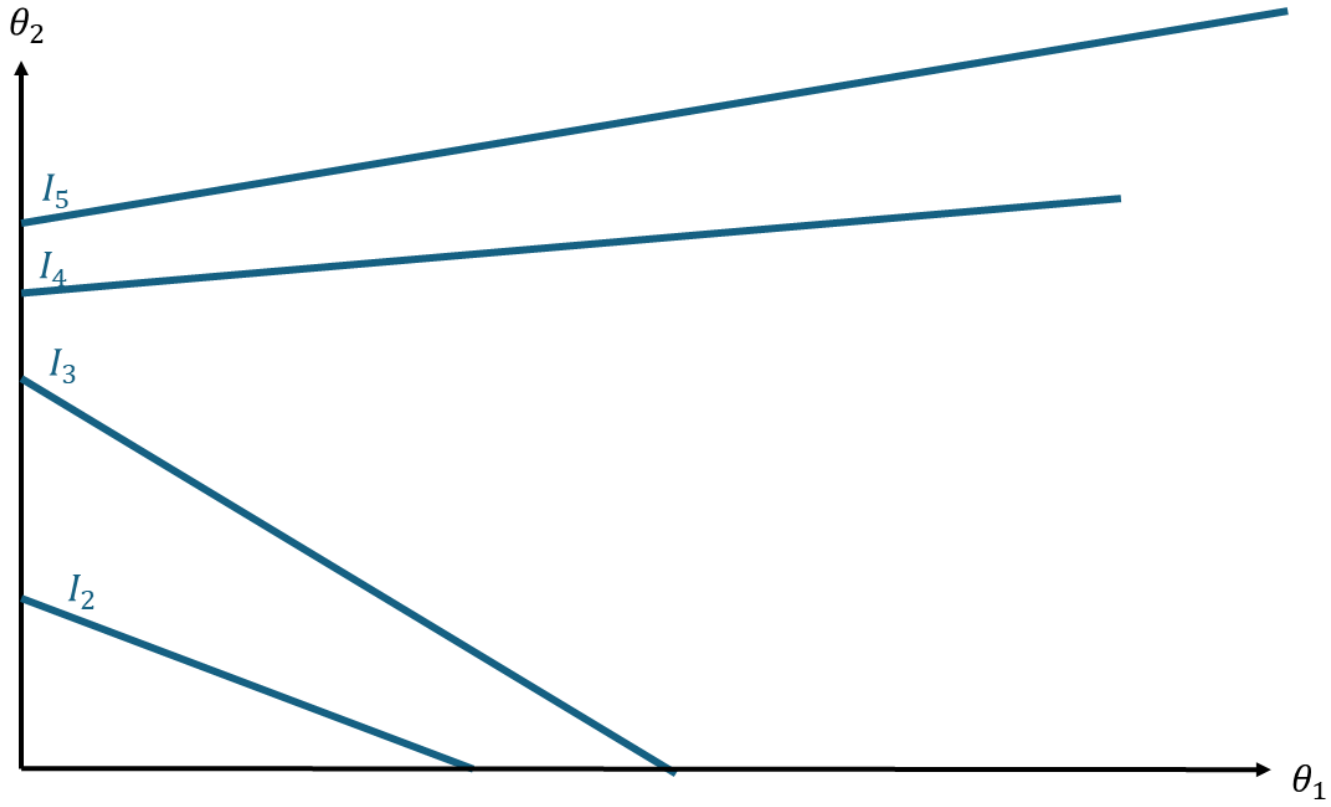


Multi-product monopoly **(proof sketch)**



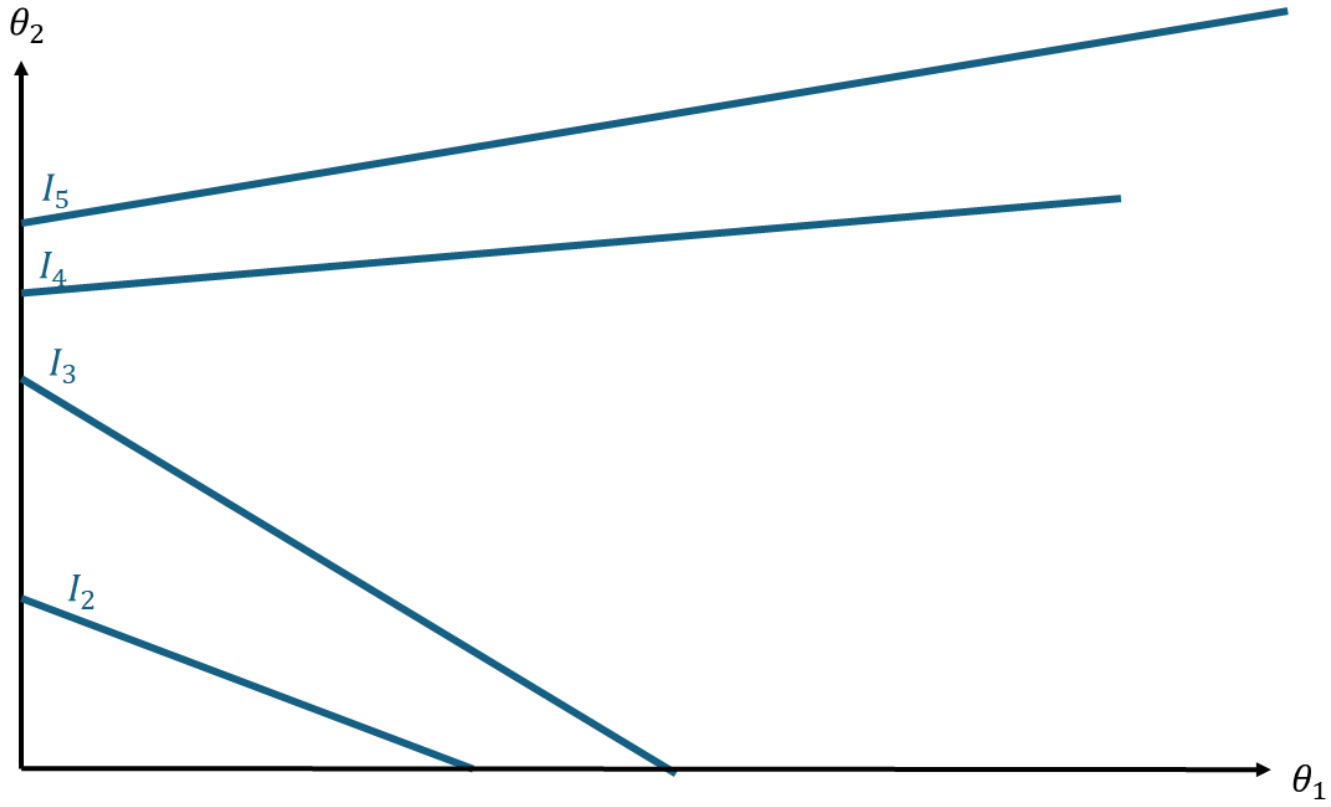
Claim: $\Delta p^k \geq 0$ for all k.

Multi-product monopoly **(proof sketch)**



Intuitively: Searchability for type 0 $\implies$ p^k is quasi-convex.

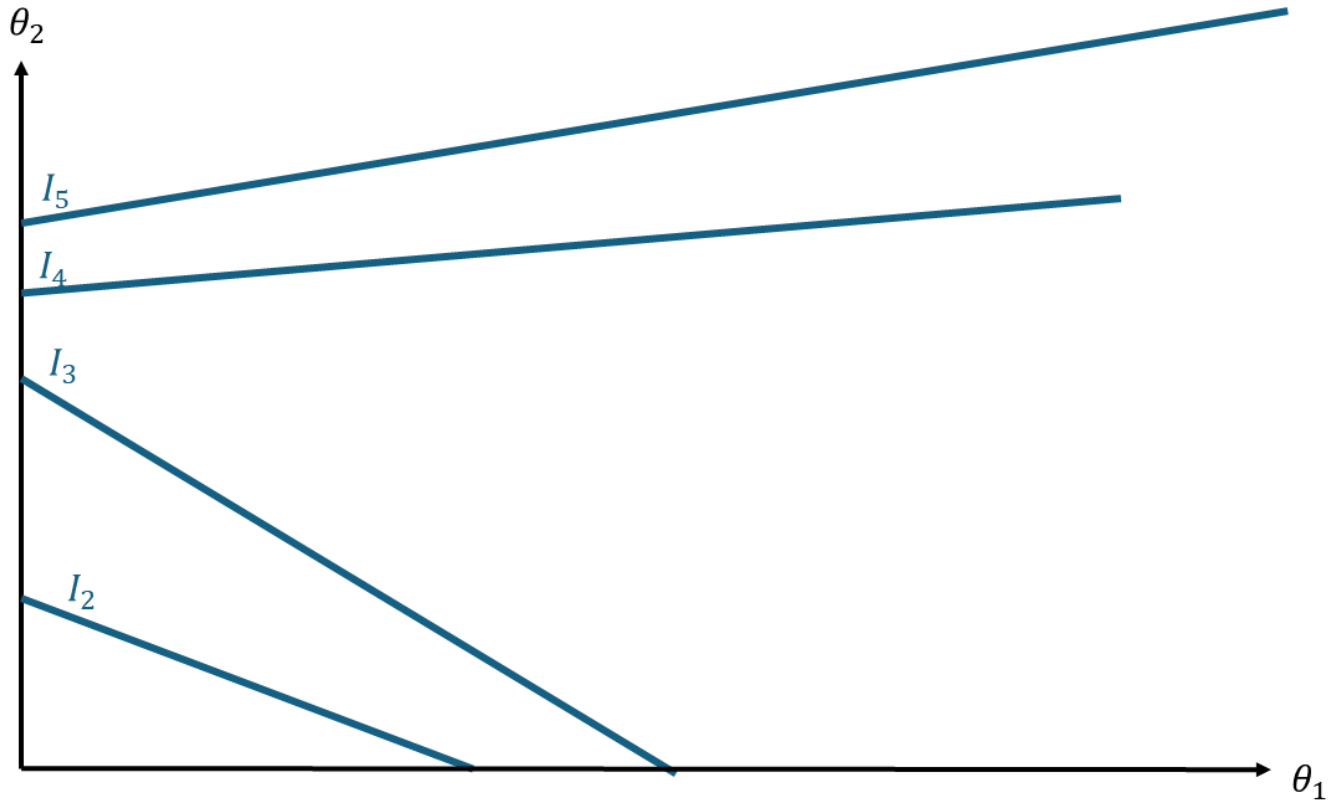
Multi-product monopoly **(proof sketch)**



Intuitively: Searchability for type 0 $\implies$ p^k is quasi-convex.

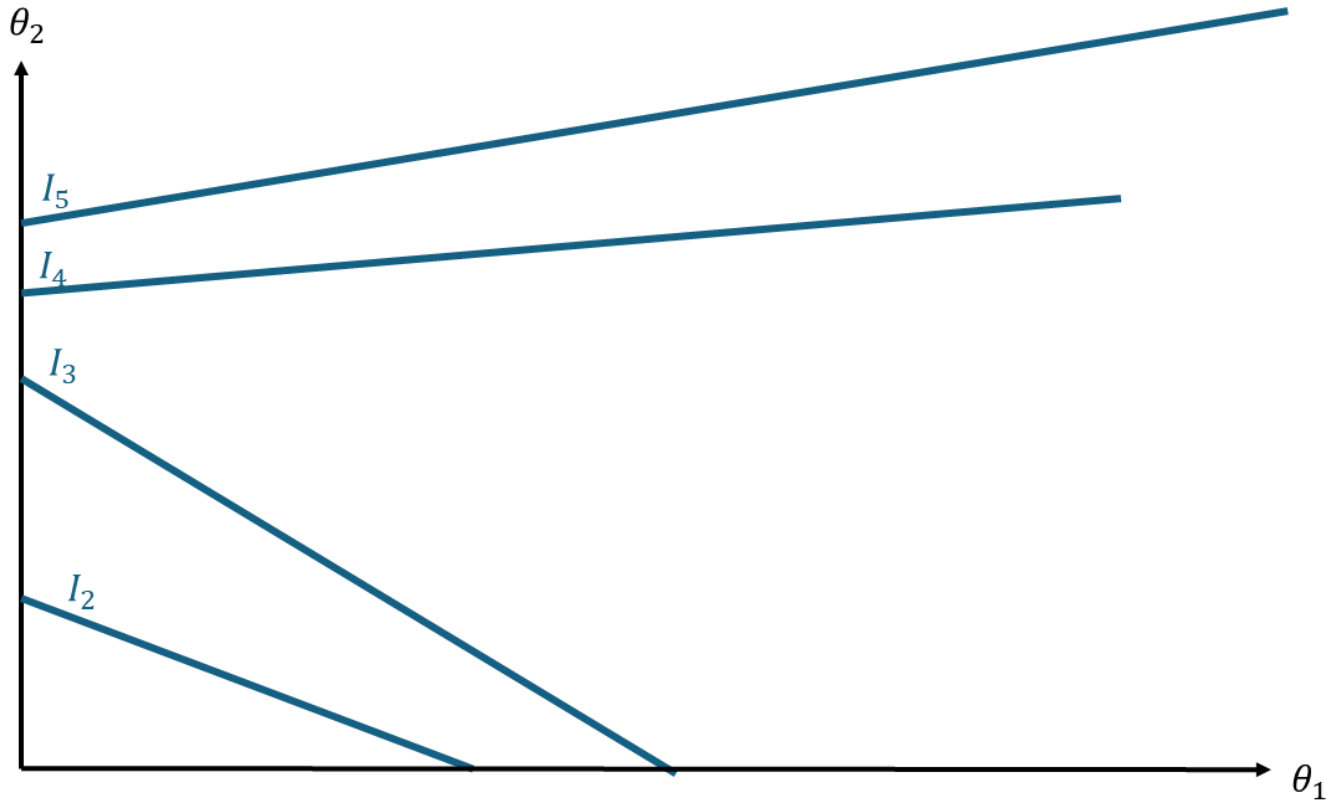
We can "chop off" negative prices.

Multi-product monopoly **(proof sketch)**



Claim: $\Delta q^k \geq 0$ for all k.

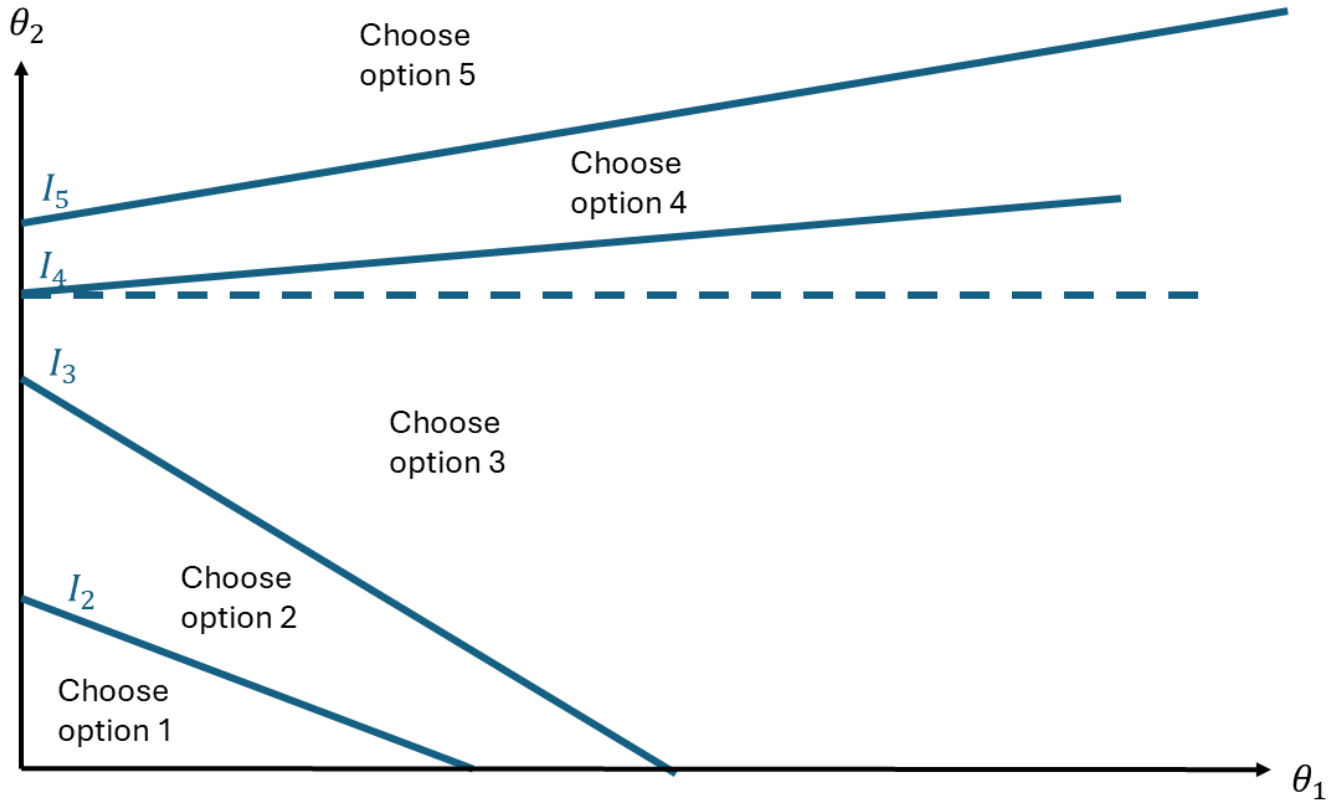
Multi-product monopoly **(proof sketch)**



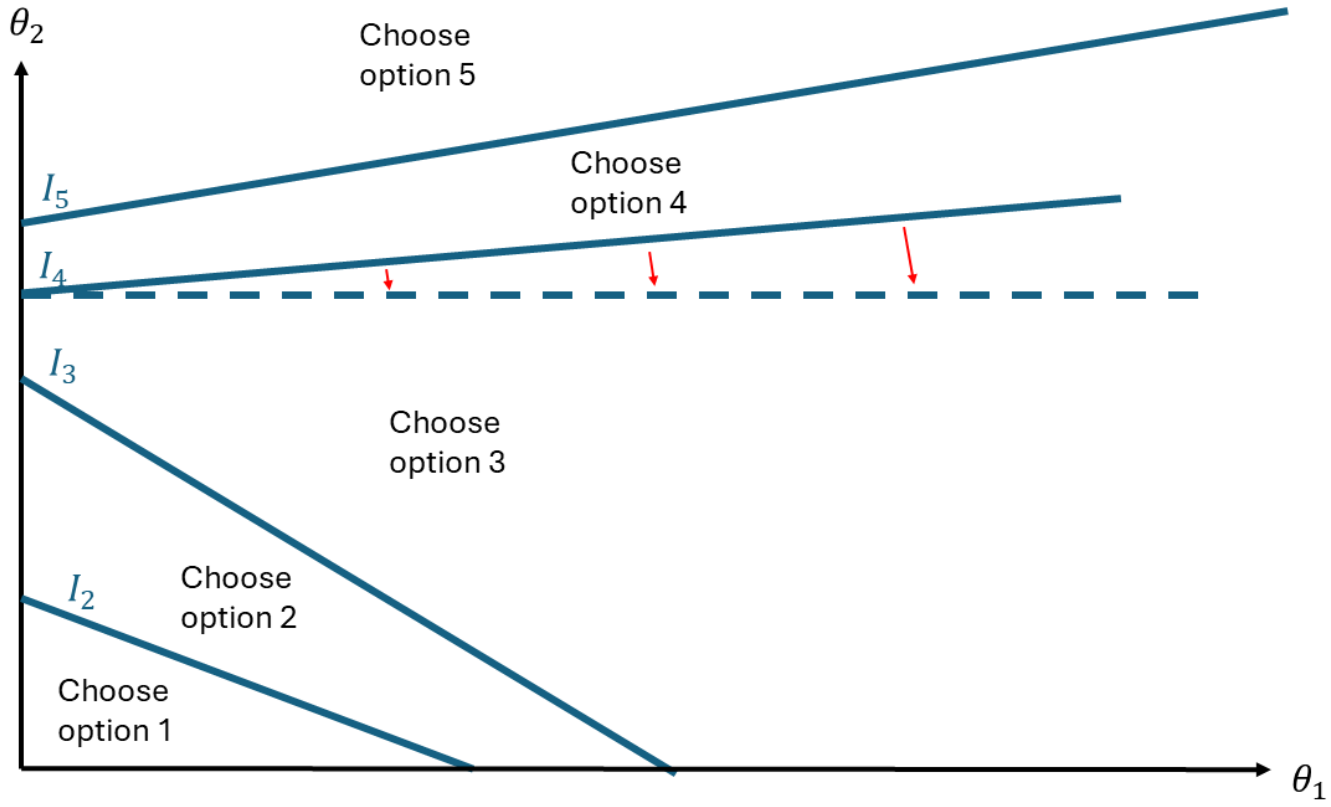
Negative slope of $I^k \iff \Delta q_1^k,\ \Delta q_2^k \geq 0$

Positive slope of $I^k \iff \Delta q_1^k \leq 0 \leq \Delta q_2^k$.

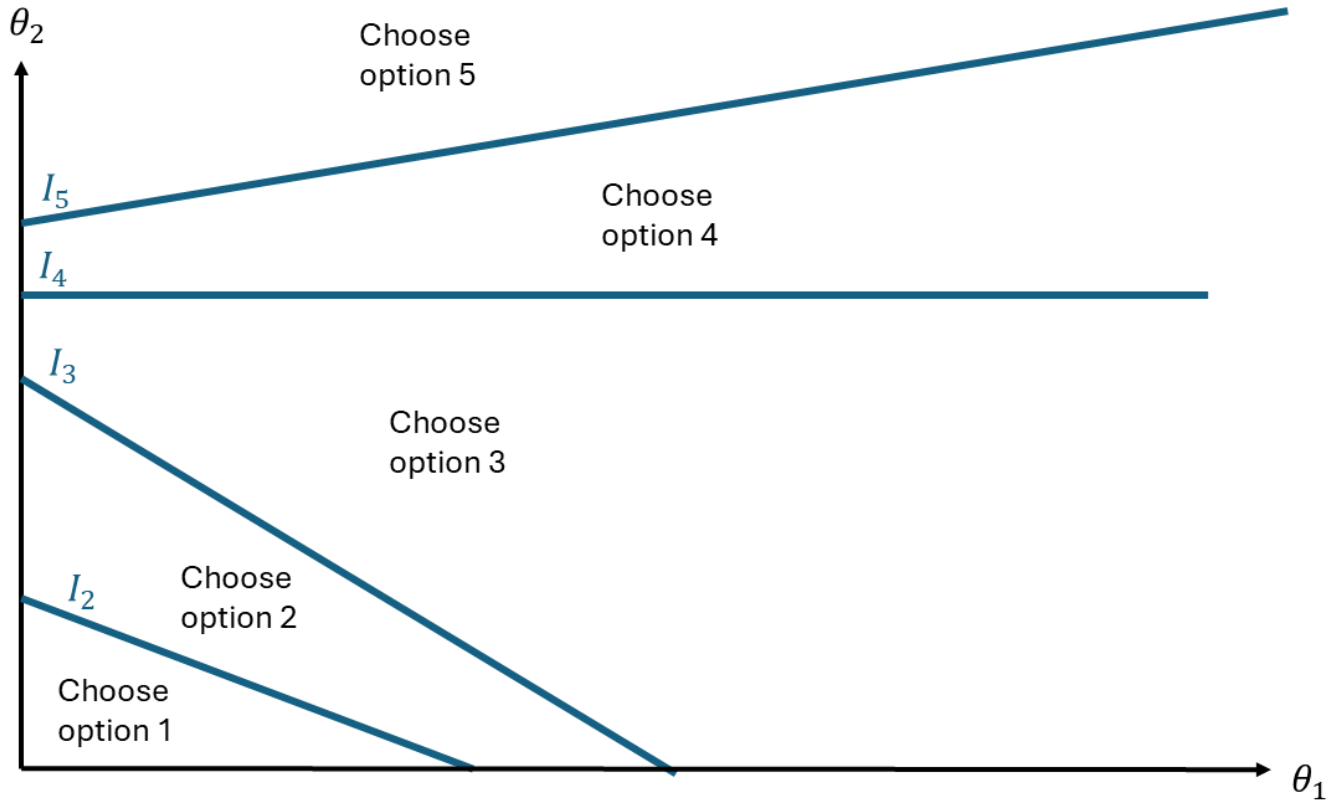
Multi-product monopoly **(proof sketch)**



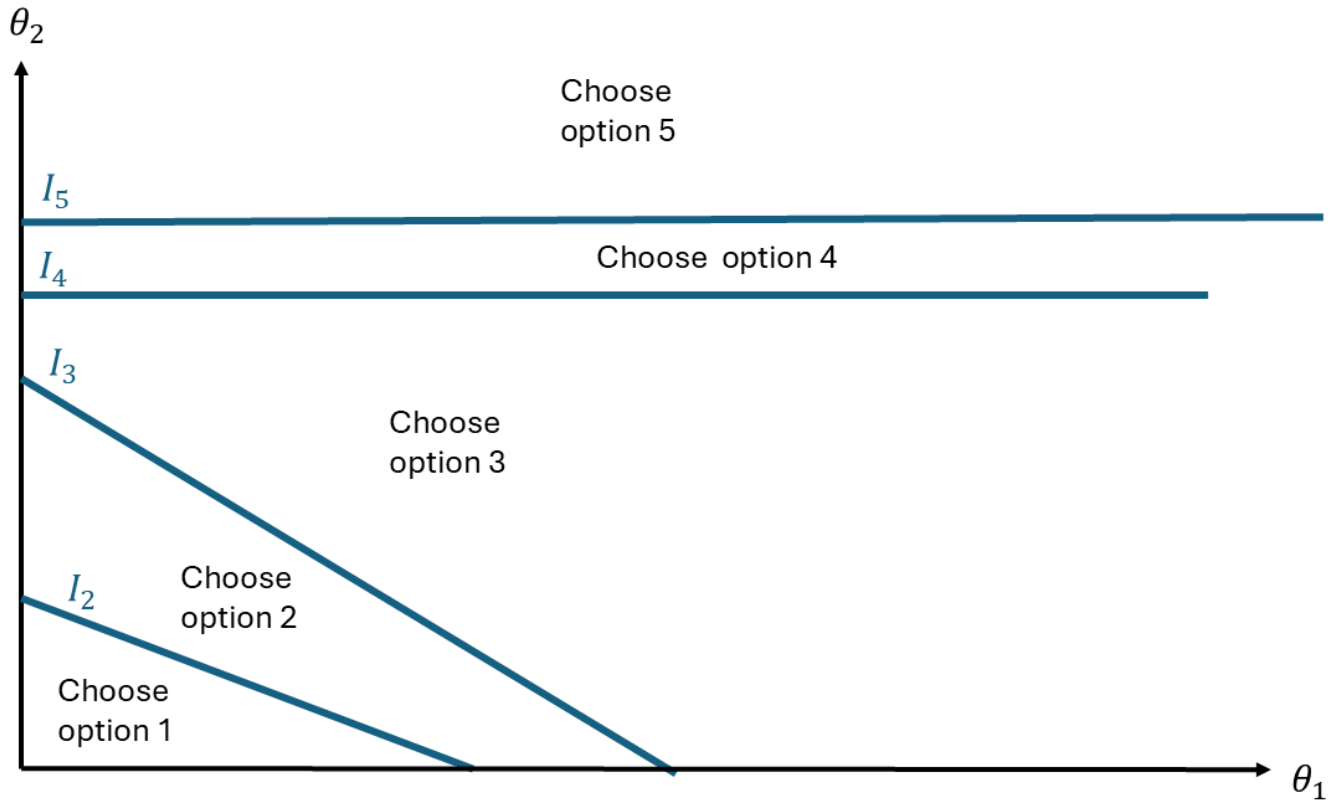
Multi-product monopoly **(proof sketch)**



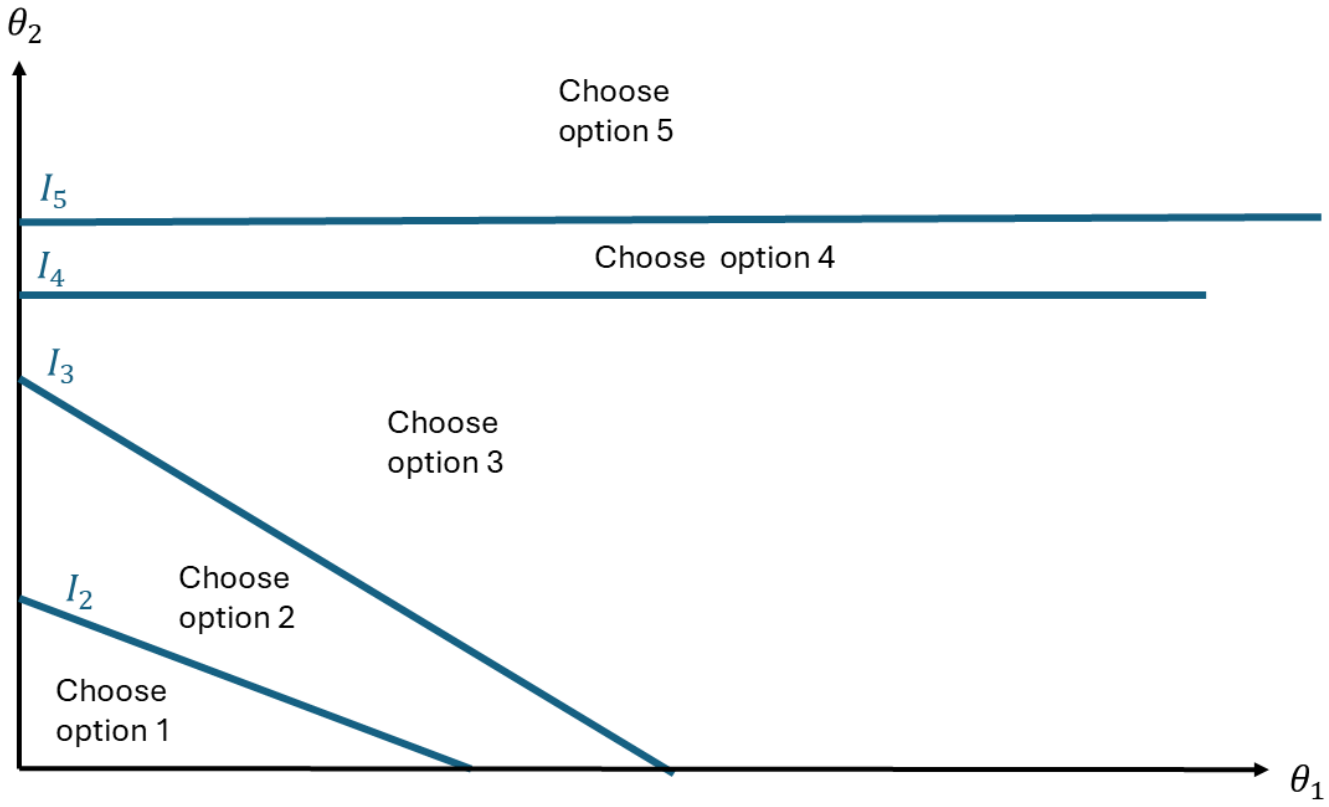
Multi-product monopoly **(proof sketch)**



Multi-product monopoly **(proof sketch)**

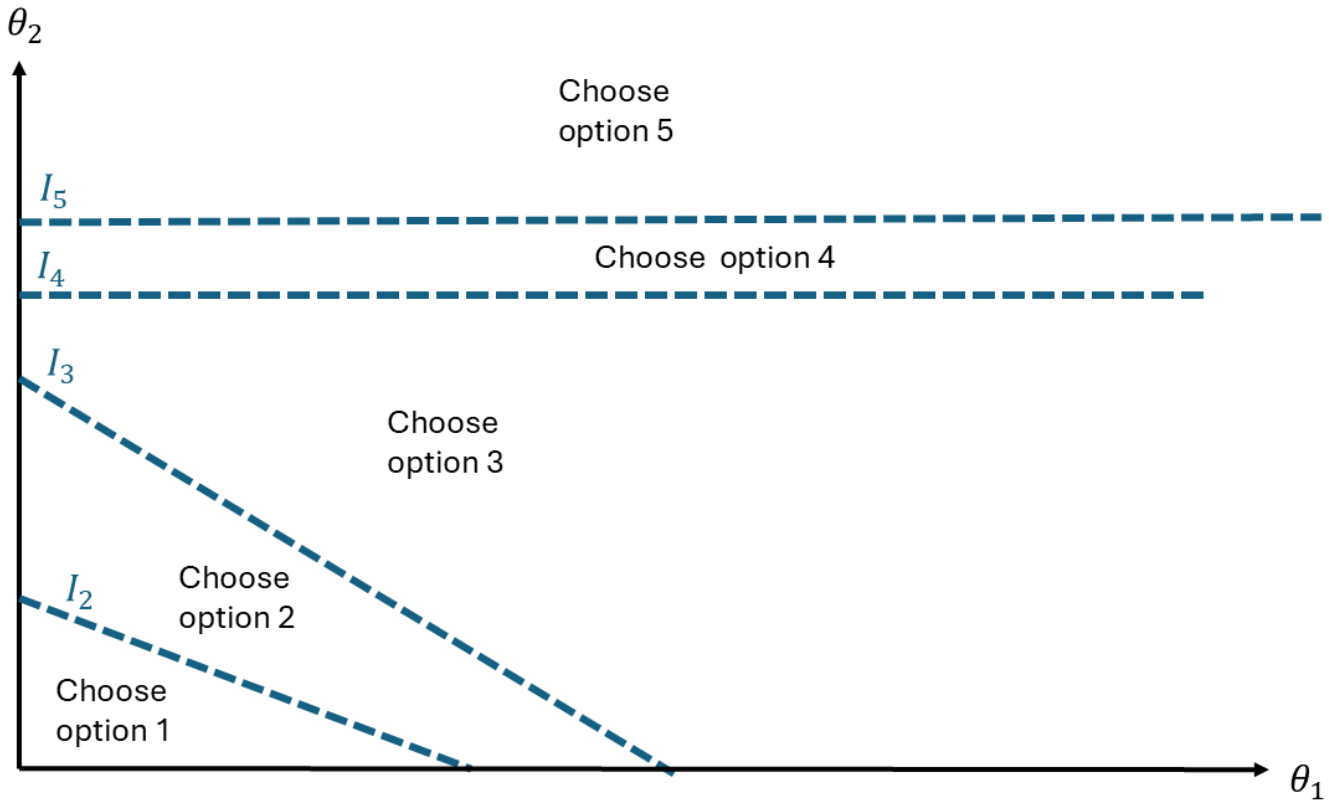


Multi-product monopoly **(proof sketch)**



Thus, it is wlog to have $\Delta q^k \geq 0$ for all k.

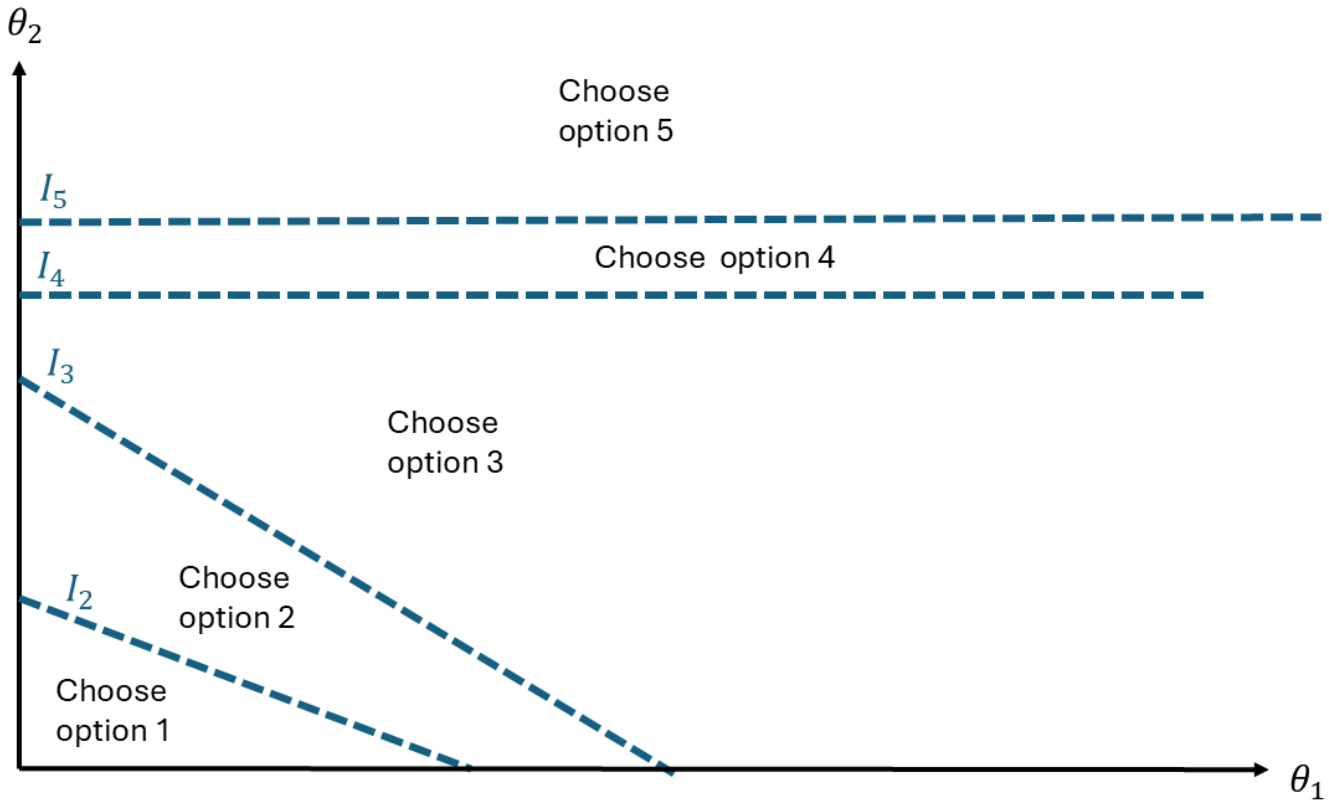
Multi-product monopoly **(proof sketch)**



Keeping I^k fixed, we can optimize over weights $\lambda(k) \geq 0$:

$$\Delta\hat{p}^k = \lambda(k)\Delta p^k \text{ and } \Delta\hat{q}^k = \lambda(k)\Delta q^k$$

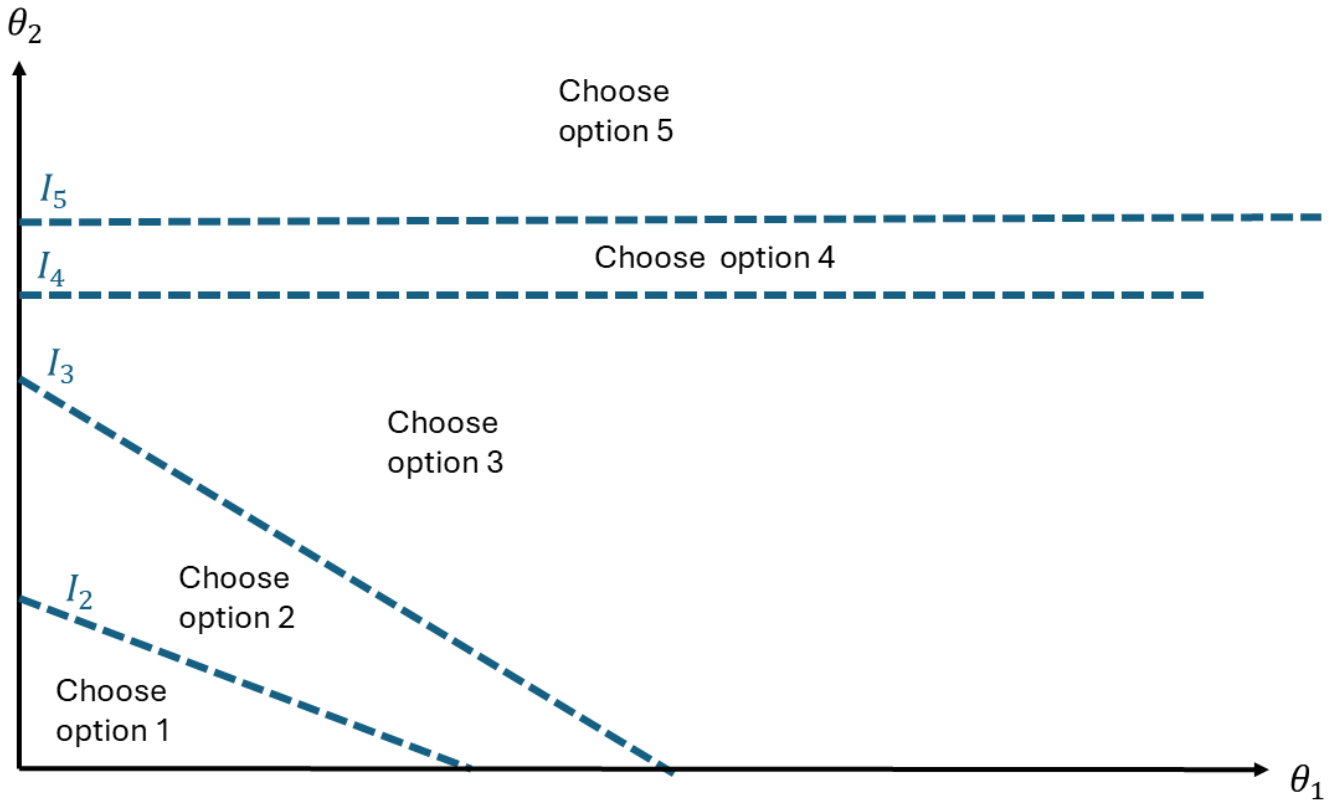
Multi-product monopoly **(proof sketch)**



Because $\Delta q^k \geq 0$ for all k, the constraint on $\lambda(k)$'s:

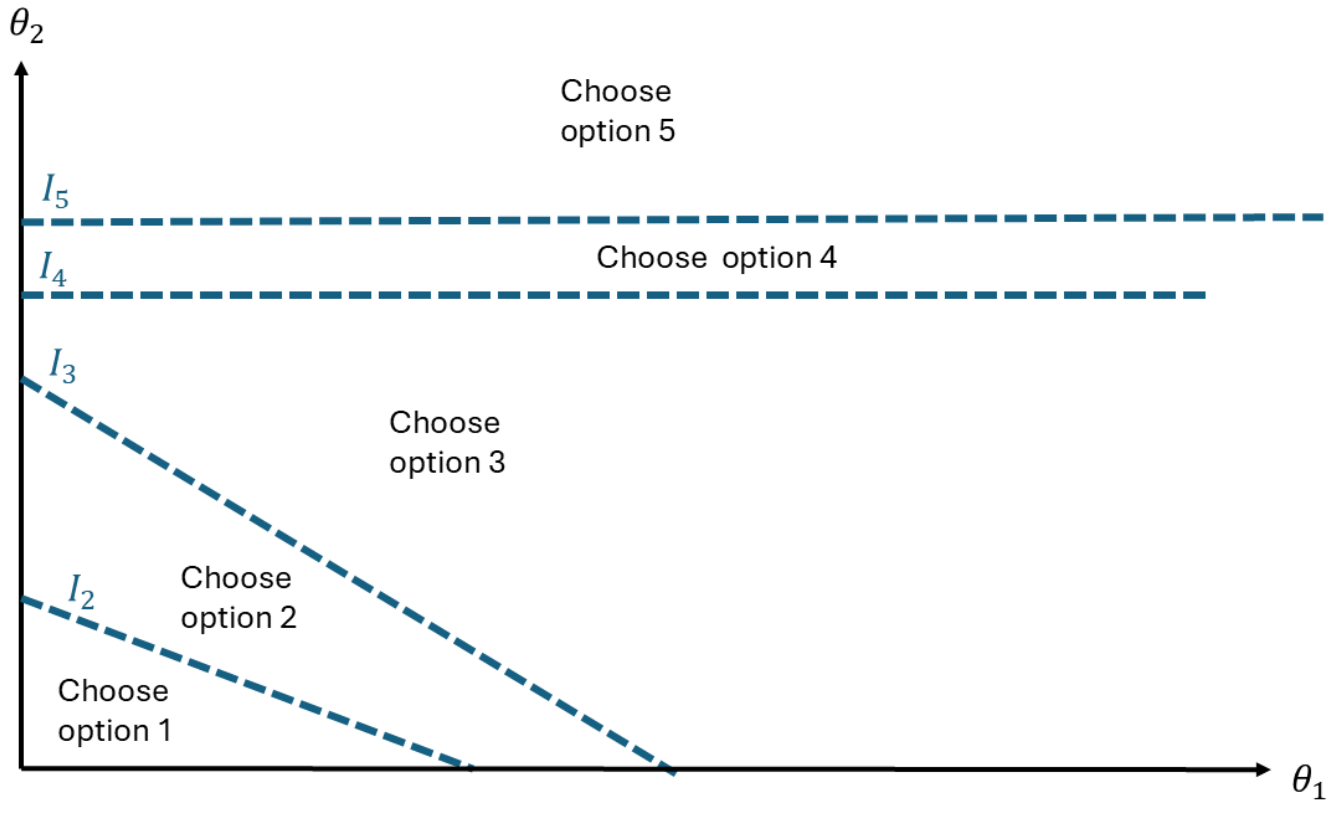
$$\sum_{k=1}^{m} \lambda(k) \Delta q_i^k \leq 1, \text{ for all } i.$$

Multi-product monopoly **(proof sketch)**



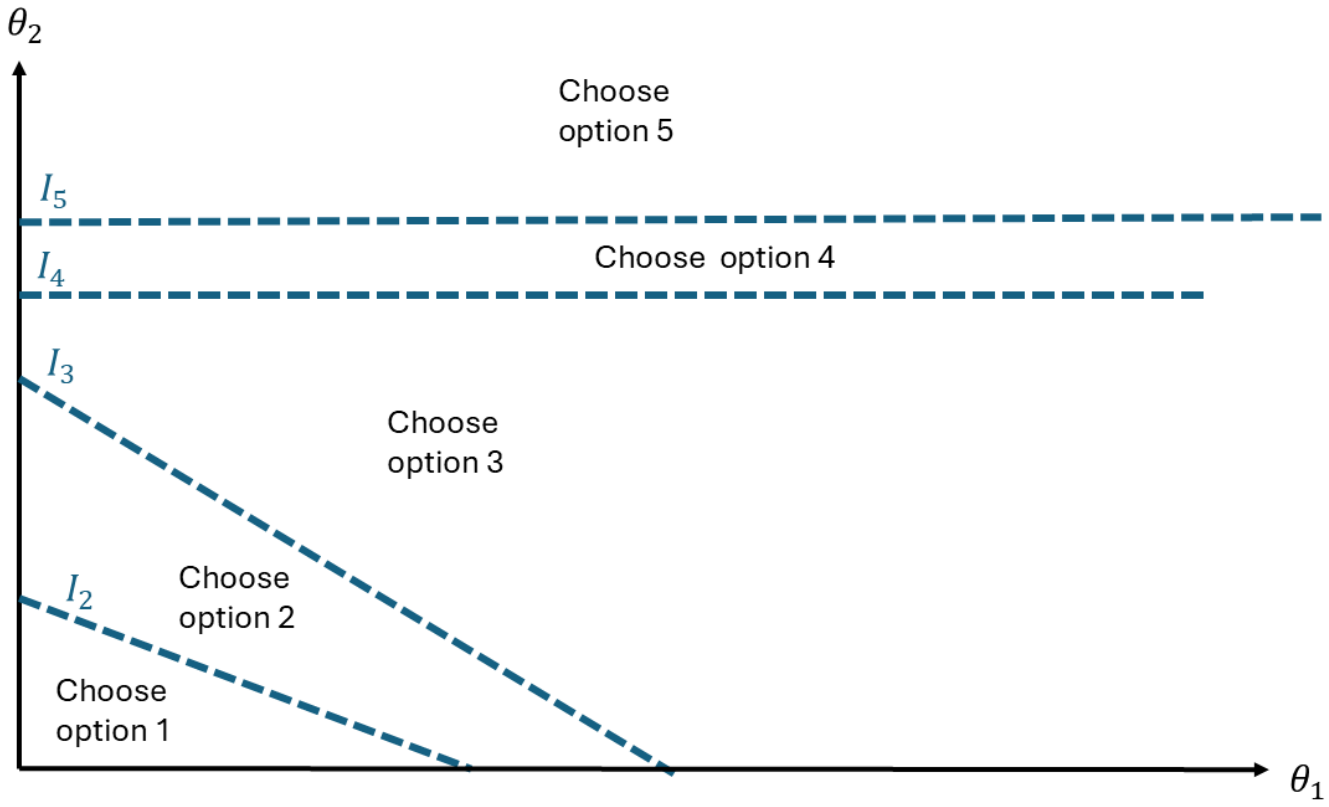
Revenue is linear in the $\lambda(k)$'s.

Multi-product monopoly **(proof sketch)**



This gives us a linear program with *n* linear constraints.

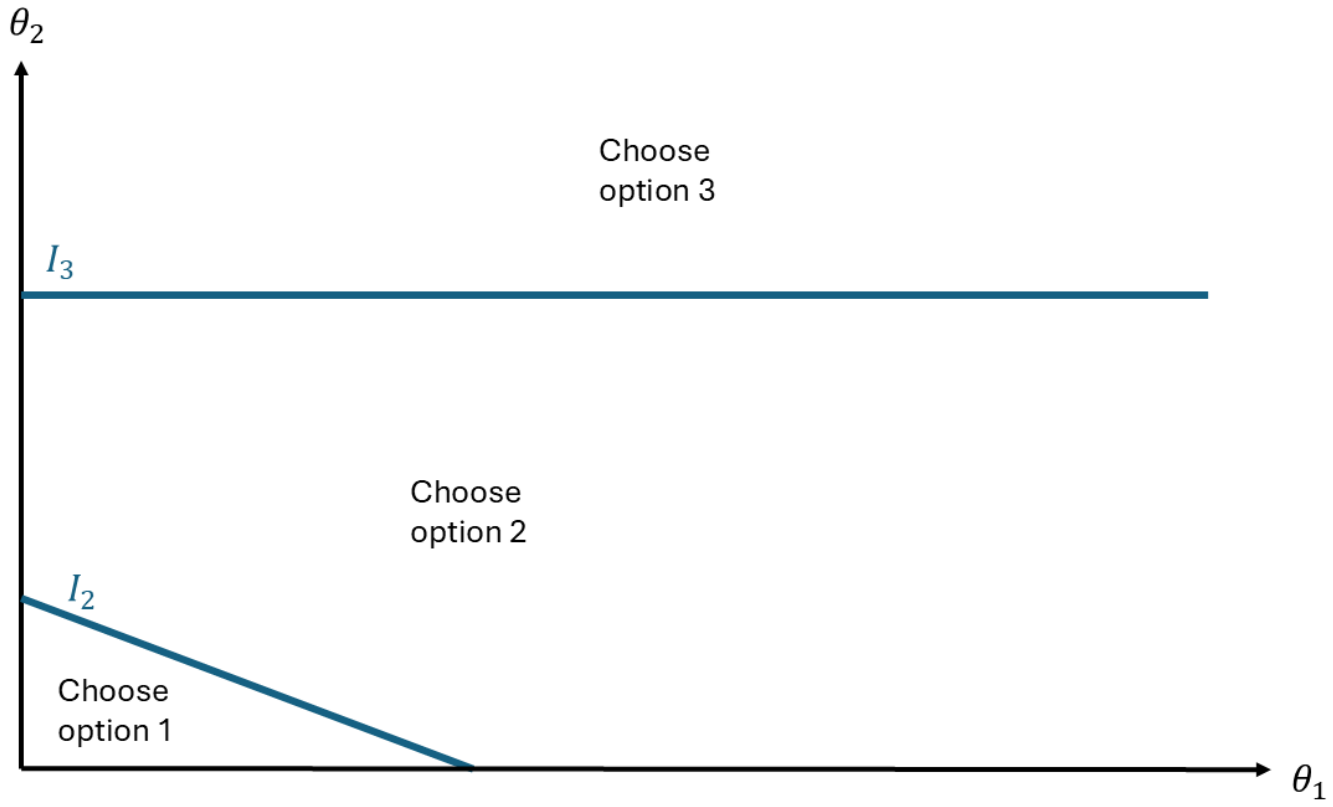
Multi-product monopoly **(proof sketch)**



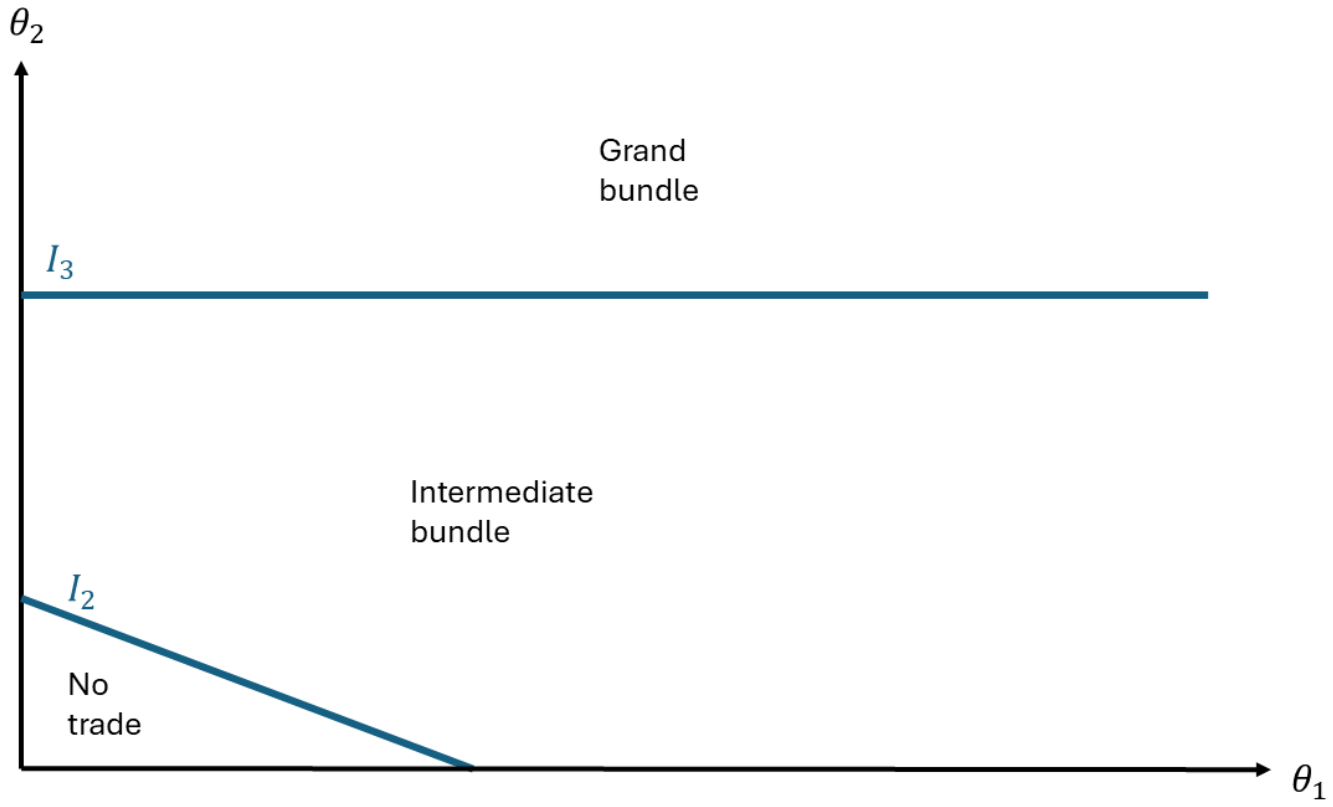
This gives us a linear program with n linear constraints.

$\implies$ only n of the $\lambda(k)$'s are non-zero.

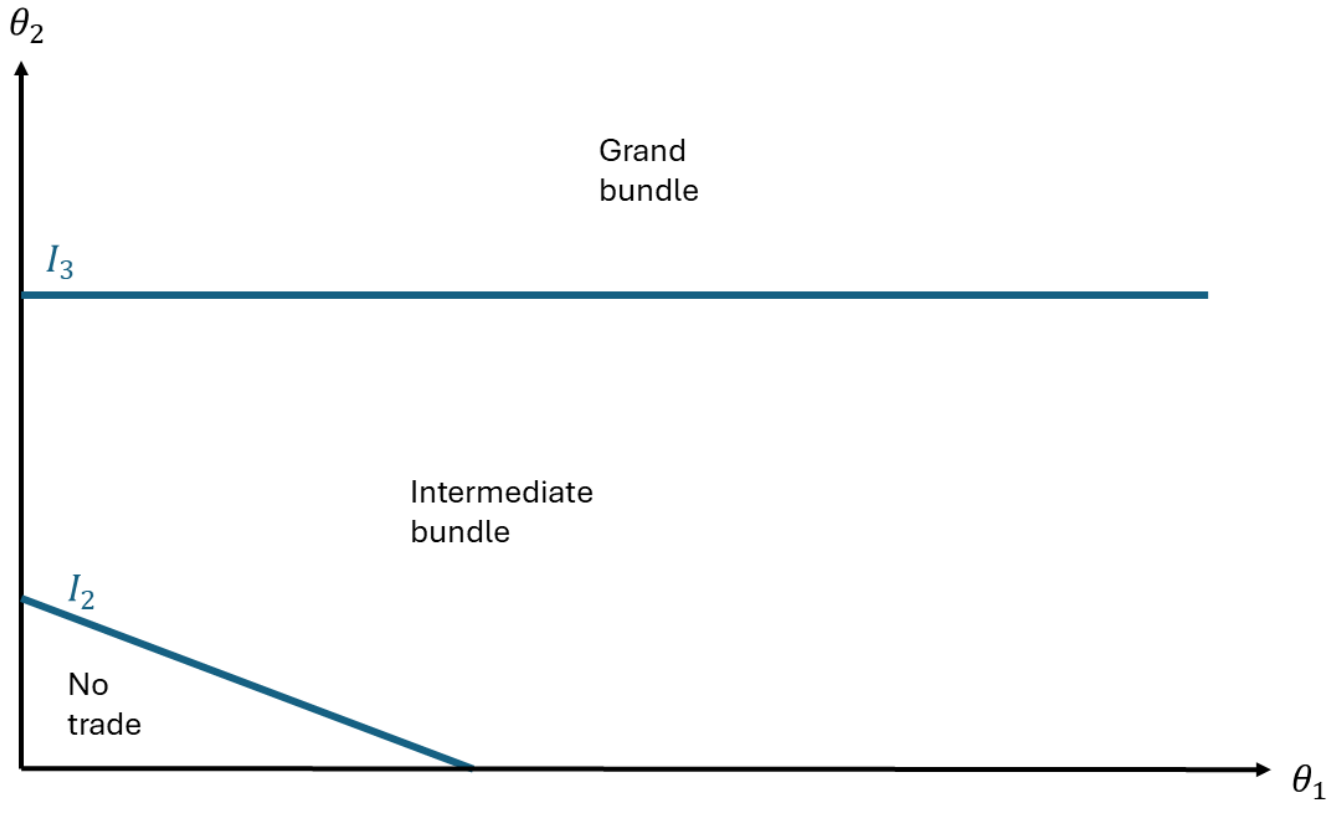
Multi-product monopoly **(proof sketch)**



Multi-product monopoly **(proof sketch)**

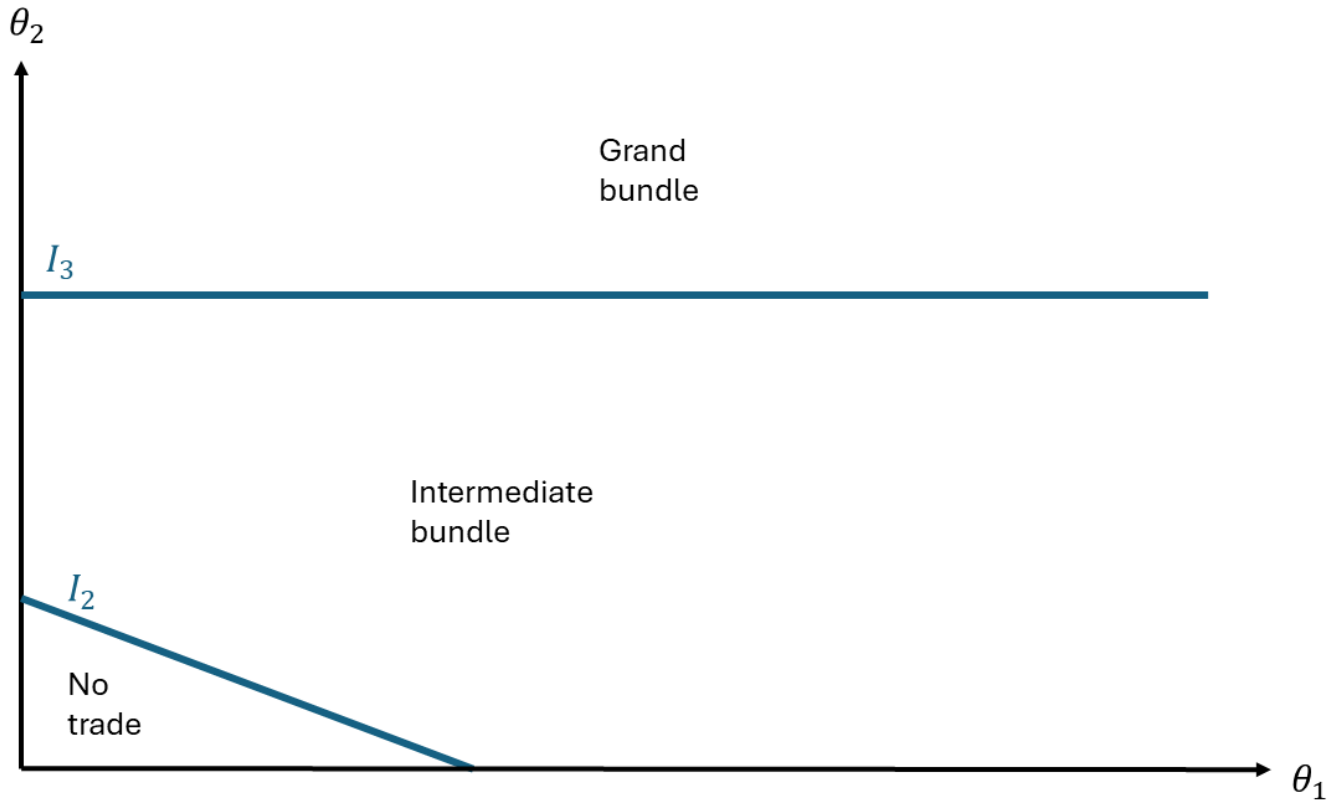


Multi-product monopoly **(proof sketch)**



The last step in the proof is to show the result for infinite menus.

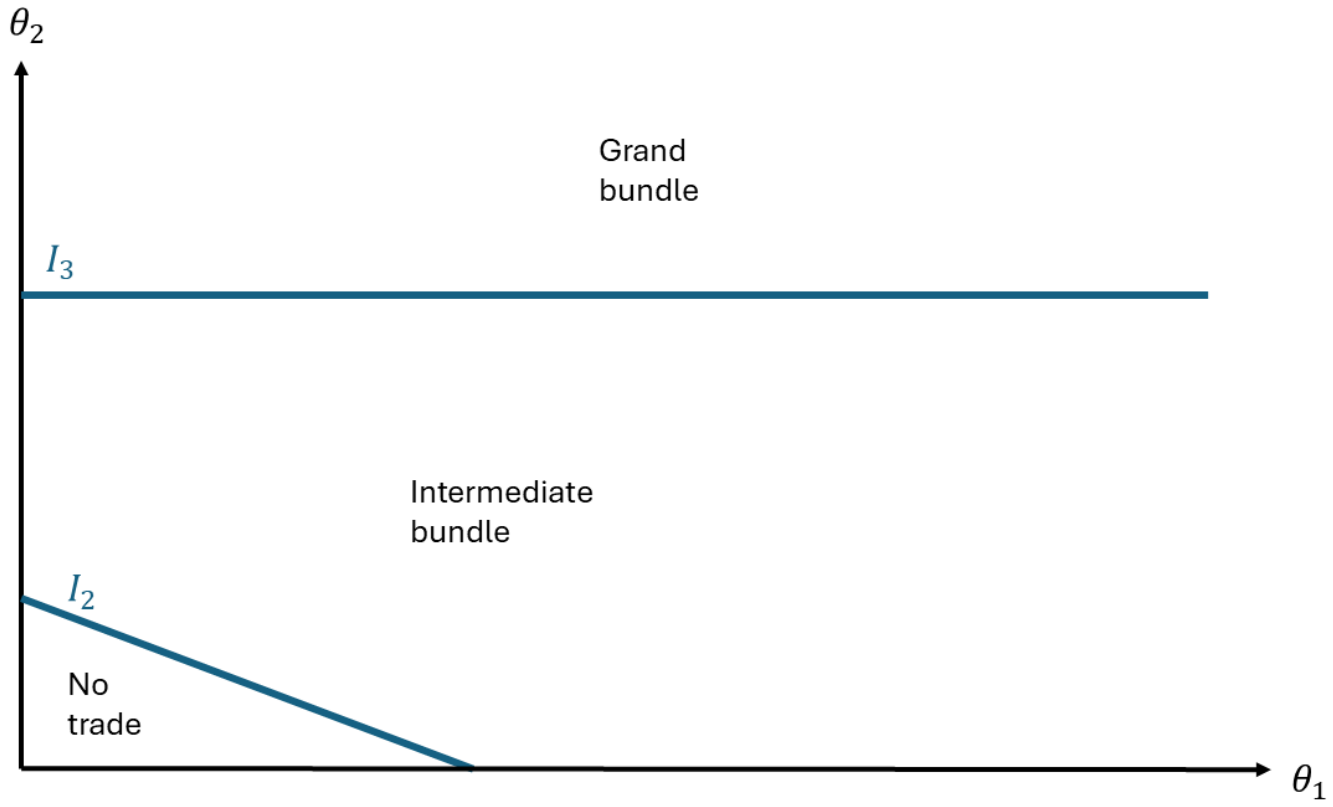
Multi-product monopoly **(proof sketch)**



The last step in the proof is to show the result for infinite menus.

This is done via an approximation argument.

Multi-product monopoly **(proof sketch)**



The last step in the proof is to show the result for infinite menus.

This is done via an approximation argument. ■

Application 2: Money and ordeals

Application 2: Money and ordeals

- Single good with quality $q \in [0, 1]$, marginal cost $C \geq 0$.

Application 2: Money and ordeals

- Single good with quality $q \in [0, 1]$, marginal cost $C \geq 0$.
- Designer screens with **payments** $p \in \mathbb{R}_+$ and **ordeals** $y \in \mathbb{R}_+^n$.

Application 2: Money and ordeals

- Single good with quality $q \in [0, 1]$, marginal cost $C \geq 0$.
- Designer screens with **payments** $p \in \mathbb{R}_+$ and **ordeals** $y \in \mathbb{R}_+^n$.
- Agent payoff $vq - p - c \cdot y$; types $(v, c) \in \mathbb{R}_+^{n+1}$ with continuous full-support density.

Application 2: Money and ordeals

- Single good with quality $q \in [0, 1]$, marginal cost $C \geq 0$.
- Designer screens with **payments** $p \in \mathbb{R}_+$ and **ordeals** $y \in \mathbb{R}_+^n$.
- Agent payoff $vq - p - c \cdot y$; types $(v, c) \in \mathbb{R}_+^{n+1}$ with continuous full-support density.
- Designer maximizes α**-welfare**, $\alpha \in [0, 1]$:

$$\alpha\, \mathbb{E}\big[w(vq - p - c \cdot y)\big] + (1 - \alpha)\, \mathbb{E}\big[p - Cq\big],$$

where $w \geq 0$ has bounded support, jointly distributed with (v, c).

Application 2: Money and ordeals

- Single good with quality $q \in [0, 1]$, marginal cost $C \geq 0$.
- Designer screens with **payments** $p \in \mathbb{R}_+$ and **ordeals** $y \in \mathbb{R}_+^n$.
- Agent payoff $vq - p - c \cdot y$; types $(v, c) \in \mathbb{R}_+^{n+1}$ with continuous full-support density.
- Designer maximizes α**-welfare**, $\alpha \in [0, 1]$:

$$\alpha\, \mathbb{E}\big[w(vq - p - c \cdot y)\big] + (1 - \alpha)\, \mathbb{E}\big[p - Cq\big],$$

  where $w \geq 0$ has bounded support, jointly distributed with (v, c).
- **Posted-requirement menu:** single $(1, p^\star, y^\star)$ (plus outside option).

Application 2: Money and ordeals

Theorem

There exists an optimal searchable menu that is a ***posted-requirement menu****.*

If the principal cares only about ***profit*** *($\alpha = 0$), then a* ***posted-price menu*** *is optimal among searchable menus.*

Application 2: Money and ordeals

Theorem

There exists an optimal searchable menu that is a ***posted-requirement menu****.*

If the principal cares only about ***profit*** *($\alpha = 0$), then a* ***posted-price menu*** *is optimal among searchable menus.*

Comments:

- Optimal mechanism is **binary**, regardless of the number of potential ordeals.

Application 2: Money and ordeals

Theorem

There exists an optimal searchable menu that is a ***posted-requirement menu****.*

If the principal cares only about ***profit*** *($\alpha = 0$), then a* ***posted-price menu*** *is optimal among searchable menus.*

Comments:

- Optimal mechanism is **binary**, regardless of the number of potential ordeals.
- Without searchability: ordeals can be optimal even for a pure-profit designer.

Application 2: Money and ordeals

Theorem

There exists an optimal searchable menu that is a ***posted-requirement menu****.*

If the principal cares only about ***profit*** *($\alpha = 0$), then a* ***posted-price menu*** *is optimal among searchable menus.*

Comments:

- Optimal mechanism is **binary**, regardless of the number of potential ordeals.
- Without searchability: ordeals can be optimal even for a pure-profit designer.
- Proof: Similar idea but **no upper bound** on y!

Application 3: Income taxation

Application 3: Income taxation

- Types $\theta \in \Theta$ (abstract space); earning $y \in [0, \bar{y}]$; tax $T(y) \in \mathbb{R}$.

Application 3: Income taxation

- Types $\theta \in \Theta$ (abstract space); earning $y \in [0, \bar{y}]$; tax $T(y) \in \mathbb{R}$.
- Agent payoff $u(c, y, \theta)$ is monotone and concave (c is consumption, equal to post-tax earnings).

Application 3: Income taxation

- Types $\theta \in \Theta$ (abstract space); earning $y \in [0, \bar{y}]$; tax $T(y) \in \mathbb{R}$.
- Agent payoff $u(c, y, \theta)$ is monotone and concave (c is consumption, equal to post-tax earnings).
- Wlog: $T(y') - T(y) < y' - y$ for $y < y'$.

Application 3: Income taxation

- Types $\theta \in \Theta$ (abstract space); earning $y \in [0, \bar{y}]$; tax $T(y) \in \mathbb{R}$.
- Agent payoff $u(c, y, \theta)$ is monotone and concave (c is consumption, equal to post-tax earnings).
- Wlog: $T(y') - T(y) < y' - y$ for $y < y'$.
- A tax schedule is **progressive** if $T(\cdot)$ is convex.

Application 3: Income taxation

- Types $\theta \in \Theta$ (abstract space); earning $y \in [0, \bar{y}]$; tax $T(y) \in \mathbb{R}$.
- Agent payoff $u(c, y, \theta)$ is monotone and concave (c is consumption, equal to post-tax earnings).
- Wlog: $T(y') - T(y) < y' - y$ for $y < y'$.
- A tax schedule is **progressive** if $T(\cdot)$ is convex.
- Type θ has **linear preferences** if payoff is $c - \alpha y$ for some $\alpha \in \mathbb{R}_+$; Θ is **rich** if it includes all linear preferences.

Application 3: Income taxation

Theorem

Any progressive tax schedule is a ***searchable menu*** *ordered by earnings.*

Conversely, if Θ *is rich, then any searchable menu must correspond to a* ***progressive tax schedule****.*

Application 3: Income taxation

Theorem

Any progressive tax schedule is a ***searchable menu*** *ordered by earnings.*

Conversely, if Θ is rich, then any searchable menu must correspond to a ***progressive tax schedule****.*

Comments:

- **Progressivity** $\iff$ **searchability** on rich type spaces.

Application 3: Income taxation

Theorem

Any progressive tax schedule is a ***searchable menu*** *ordered by earnings.*

Conversely, if Θ is rich, then any searchable menu must correspond to a ***progressive tax schedule****.*

Comments:

- **Progressivity** $\iff$ **searchability** on rich type spaces.
- Provides a **normative foundation** for progressive taxation (see also Doligalski, 2019 and Vairo, 2025).

Application 3: Income taxation

Theorem

*Any progressive tax schedule is a **searchable menu** ordered by earnings.*

*Conversely, if Θ is rich, then any searchable menu must correspond to a **progressive tax schedule**.*

Comments:

- **Progressivity** $\iff$ **searchability** on rich type spaces.
- Provides a **normative foundation** for progressive taxation (see also Doligalski, 2019 and Vairo, 2025).
- Searchability is a necessary condition for the **tax perturbation** approach (Saez, 2001) to work!

Extension: Grid searchability

Extension: Grid searchability

- Note that searchability in the multi-product monopoly problem with additive valuations **rules out** additive pricing.

Extension: Grid searchability

- Note that searchability in the multi-product monopoly problem with additive valuations **rules out** additive pricing.

- Bigger problem: In a taxation problem with multi-dimensional allocation, “laissez-faire” is **not searchable**.

Extension: Grid searchability

- Note that searchability in the multi-product monopoly problem with additive valuations **rules out** additive pricing.

- Bigger problem: In a taxation problem with multi-dimensional allocation, “laissez-faire” is **not searchable**.

- Intuitively, agent’s problem is still tractable if it decomposes into a small number of separable decisions.

Extension: Grid searchability

- Note that searchability in the multi-product monopoly problem with additive valuations **rules out** additive pricing.
- Bigger problem: In a taxation problem with multi-dimensional allocation, “laissez-faire” is **not searchable**.
- Intuitively, agent’s problem is still tractable if it decomposes into a small number of separable decisions.
- We will relax our constraint by allowing choices from **multiple menus**.

Grid searchability: Environment

- Allocation space $Q = \prod_{i=1}^{n} Q_i$, where each $Q_i \subseteq \mathbb{R}$ is an interval.

Grid searchability: Environment

- Allocation space $Q = \prod_{i=1}^{n} Q_i$, where each $Q_i \subseteq \mathbb{R}$ is an interval.
- Quasilinear, **additively separable** preferences:

$$u((q,p),\theta) = v(q,\theta) - p, \qquad v(q,\theta) = \sum_{i=1}^{n} v_i(q_i,\theta).$$

Grid searchability: Environment

- Allocation space $Q = \prod_{i=1}^{n} Q_i$, where each $Q_i \subseteq \mathbb{R}$ is an interval.
- Quasilinear, **additively separable** preferences:

$$u((q,p),\theta) = v(q,\theta) - p, \qquad v(q,\theta) = \sum_{i=1}^{n} v_i(q_i,\theta).$$

- Assume that v is **nondecreasing and concave** in q.

Grid searchability: Environment

- Allocation space $Q = \prod_{i=1}^{n} Q_i$, where each $Q_i \subseteq \mathbb{R}$ is an interval.
- Quasilinear, **additively separable** preferences:

$$u((q,p),\theta) = v(q,\theta) - p, \qquad v(q,\theta) = \sum_{i=1}^{n} v_i(q_i,\theta).$$

- Assume that v is **nondecreasing and concave** in q.
- The type space is **rich** if it includes all linear preferences:

$$v(q,\theta) = \alpha \cdot q, \qquad \alpha \in \mathbb{R}_+^n.$$

Grid searchability: Environment

- Allocation space $Q = \prod_{i=1}^{n} Q_i$, where each $Q_i \subseteq \mathbb{R}$ is an interval.
- Quasilinear, **additively separable** preferences:

$$u((q,p),\theta) = v(q,\theta) - p, \qquad v(q,\theta) = \sum_{i=1}^{n} v_i(q_i,\theta).$$

- Assume that v is **nondecreasing and concave** in q.
- The type space is **rich** if it includes all linear preferences:

$$v(q,\theta) = \alpha \cdot q, \qquad \alpha \in \mathbb{R}^n_+.$$

- We restrict attention to **finite menus**.

Grid searchability: Product menus

- Fix a partition $\mathcal{K}$ of $\{1, \ldots, n\}$.

Grid searchability: Product menus

- Fix a partition $\mathcal{K}$ of $\{1, \ldots, n\}$.
- A **product menu** $(\mathcal{Y}, p, \preceq)$ consists of:
  - a product of finite component menus,

$$\mathcal{Y} = \prod_{K \in \mathcal{K}} Y_K, \qquad Y_K \subseteq \prod_{i \in K} Q_i;$$

  - an order $\preceq_K$ on each component menu Y_K;
  - a price function $p : \mathcal{Y} \to \mathbb{R}$.

Grid searchability: Product menus

- Fix a partition $\mathcal{K}$ of $\{1, \ldots, n\}$.
- A **product menu** $(\mathcal{Y}, p, \preceq)$ consists of:
  - a product of finite component menus,

$$\mathcal{Y} = \prod_{K \in \mathcal{K}} Y_K, \qquad Y_K \subseteq \prod_{i \in K} Q_i;$$

  - an order $\preceq_K$ on each component menu Y_K;
  - a price function $p : \mathcal{Y} \to \mathbb{R}$.
- The agent's payoff is $u_\theta(y) = v(y, \theta) - p(y)$.

Grid searchability: Product menus

- Fix a partition $\mathcal{K}$ of $\{1, \ldots, n\}$.
- A **product menu** $(\mathcal{Y}, p, \preceq)$ consists of:
  - a product of finite component menus,

$$\mathcal{Y} = \prod_{K \in \mathcal{K}} Y_K, \qquad Y_K \subseteq \prod_{i \in K} Q_i;$$

  - an order $\preceq_K$ on each component menu Y_K;
  - a price function $p : \mathcal{Y} \to \mathbb{R}$.
- The agent's payoff is $u_\theta(y) = v(y, \theta) - p(y)$.
- A **neighbor** of y changes exactly **one** component y_K to its predecessor or successor in $\preceq_K$.

Grid searchability: Definition

An allocation $y \in \mathcal{Y}$ is a **local maximum** of a function u if

$$u(y) \geq u(z) \qquad \text{for every neighbor } z \text{ of } y.$$

Grid searchability: Definition

An allocation $y \in \mathcal{Y}$ is a **local maximum** of a function u if

$$u(y) \geq u(z) \qquad \text{for every neighbor } z \text{ of } y.$$

Definition

A product menu is **grid-searchable** if, for every type, every local maximum is a global maximum.

Grid searchability: Definition

An allocation $y \in \mathcal{Y}$ is a **local maximum** of a function u if

$$u(y) \geq u(z) \qquad \text{for every neighbor } z \text{ of } y.$$

Definition

A product menu is **grid-searchable** if, for every type, every local maximum is a global maximum.

- Starting anywhere, successive strictly improving local moves reach a **global optimum**.

Grid searchability: Definition

An allocation $y \in \mathcal{Y}$ is a **local maximum** of a function u if

$$u(y) \geq u(z) \qquad \text{for every neighbor } z \text{ of } y.$$

Definition

A product menu is **grid-searchable** if, for every type, every local maximum is a global maximum.

- Starting anywhere, successive strictly improving local moves reach a **global optimum**.
- With a single component menu, every searchable menu is grid-searchable.

Grid searchability: Characterization

Say that p has ***block-additive pricing*** if there exist functions $p_K : \mathcal{Y}_K \to \mathbb{R}$ for all $K \in \mathcal{K}$ such that

$$p(y) = \sum_{K \in \mathcal{K}} p_K(y_K) \,.$$

Theorem

Let $(\mathcal{Y}, p, \preceq)$ be a product menu with a strictly increasing p.

- *If Θ is rich and $(\mathcal{Y}, p, \preceq)$ is grid-searchable, then there exists an outcome-equivalent product menu with **block-additive pricing**, where each **component menu is searchable**.*
- *If $(\mathcal{Y}, p, \preceq)$ has block-additive pricing where each component menu is searchable, then $(\mathcal{Y}, p, \preceq)$ is grid-searchable.*

Application 1: Multi-product monopoly

Return to **Application 1**: $q \in [0, 1]^n$, additive values, and a full-support type density with finite first moments.

Application 1: Multi-product monopoly

Return to **Application 1**: $q \in [0, 1]^n$, additive values, and a full-support type density with finite first moments.

Corollary

For any partition $\mathcal{K}$, there exists an optimal menu among grid-searchable menus with strictly increasing prices that has:

- ***block-additive pricing**;*
- *an **upgrade menu** within each block K, with at most $|K|$ priced tiers and the full bundle at the highest tier.*

Application 1: Multi-product monopoly

Return to **Application 1**: $q \in [0, 1]^n$, additive values, and a full-support type density with finite first moments.

Corollary

For any partition $\mathcal{K}$, there exists an optimal menu among grid-searchable menus with strictly increasing prices that has:

- ***block-additive pricing**;*
- *an **upgrade menu** within each block K, with at most $|K|$ priced tiers and the full bundle at the highest tier.*

Coarsest partition: the original upgrade menu.

Finest partition: separate posted prices for each good.

Application 2: Money and ordeals

Return to **Application 2**: Screening with money and ordeals.

Application 2: Money and ordeals

Return to **Application 2**: Screening with money and ordeals.

Corollary

For any partition and any grid-searchable product menu with strictly increasing prices, there exists a ***posted-requirement menu*** *achieving a weakly better outcome for the designer.*

Application 2: Money and ordeals

Return to **Application 2**: Screening with money and ordeals.

Corollary

For any partition and any grid-searchable product menu with strictly increasing prices, there exists a ***posted-requirement menu*** *achieving a weakly better outcome for the designer.*

- Under block-additive pricing, components containing only ordeals offer no gain relative to choosing zero ordeals.

Application 2: Money and ordeals

Return to **Application 2**: Screening with money and ordeals.

Corollary

For any partition and any grid-searchable product menu with strictly increasing prices, there exists a ***posted-requirement menu*** *achieving a weakly better outcome for the designer.*

- Under block-additive pricing, components containing only ordeals offer no gain relative to choosing zero ordeals.
- Only the component containing the productive dimension q matters $\Longrightarrow$ previous result applies.

Application 3: Multiple income sources

- Income vector $y \in [0, \bar{y}]^n$ and tax schedule $T(y)$.

Application 3: Multiple income sources

- Income vector $y \in [0, \bar{y}]^n$ and tax schedule $T(y)$.
- Preferences and budget:

$$u(c, \theta) - \sum_i h_i(y_i, \theta), \qquad c \leq \sum_i y_i - T(y).$$

Application 3: Multiple income sources

- Income vector $y \in [0, \bar{y}]^n$ and tax schedule $T(y)$.
- Preferences and budget:

$$u(c, \theta) - \sum_i h_i(y_i, \theta), \qquad c \leq \sum_i y_i - T(y).$$

- Assume that:

Application 3: Multiple income sources

- Income vector $y \in [0, \bar{y}]^n$ and tax schedule $T(y)$.
- Preferences and budget:

$$u(c, \theta) - \sum_i h_i(y_i, \theta), \qquad c \leq \sum_i y_i - T(y).$$

- Assume that:
  - Agent's search problem is over the **Lagrangian**:

$$\big(u(c, \theta) - \lambda c\big) + \sum_i \big(\lambda y_i - h_i(y_i, \theta)\big) - \lambda T(y).$$

Application 3: Multiple income sources

- Income vector $y \in [0, \bar{y}]^n$ and tax schedule $T(y)$.
- Preferences and budget:

$$u(c, \theta) - \sum_i h_i(y_i, \theta), \qquad c \leq \sum_i y_i - T(y).$$

- Assume that:
  - Agent's search problem is over the **Lagrangian**:

$$\big(u(c, \theta) - \lambda c\big) + \sum_i \big(\lambda y_i - h_i(y_i, \theta)\big) - \lambda T(y).$$

  - $T(y_i', y_{-i}) - T(y_i, y_{-i}) < y_i' - y_i$ for all $y_i' > y_i$ and y_{-i}.

Application 3: Multiple income sources

- Income vector $y \in [0, \bar{y}]^n$ and tax schedule $T(y)$.
- Preferences and budget:

$$u(c, \theta) - \sum_i h_i(y_i, \theta), \qquad c \leq \sum_i y_i - T(y).$$

- Assume that:
  - Agent's search problem is over the **Lagrangian**:

$$\big(u(c, \theta) - \lambda c\big) + \sum_i \big(\lambda y_i - h_i(y_i, \theta)\big) - \lambda T(y).$$

  - $T(y_i', y_{-i}) - T(y_i, y_{-i}) < y_i' - y_i$ for all $y_i' > y_i$ and y_{-i}.
  - **Richness:** all linear preferences $c - \sum_i \alpha_i y_i$ are present.

Application 3: Multiple income sources

Corollary

*Suppose $\mathcal{K}$ is the **finest** partition. Then,*

$$T(y) = \sum_i T_i(y_i) \text{ and each } T_i \text{ is progressive}$$

if and only if the product menu is grid-searchable (for any discrete grid of income levels).

Application 3: Multiple income sources

Corollary

*Suppose $\mathcal{K}$ is the **finest** partition. Then,*

$$T(y) = \sum_i T_i(y_i) \text{ and each } T_i \text{ is progressive}$$

if and only if the product menu is grid-searchable (for any discrete grid of income levels).

Implication: Separate search across income sources requires **separable taxation**, with **progressivity within each source**.

Note: Coarser partitions can permit nonseparable taxes.

Concluding remarks

Concluding remarks

Things that I **like** about this paper:

- A fairly natural, economically meaningful restriction makes multidimensional screening **manageable**.

Concluding remarks

Things that I **like** about this paper:

- A fairly natural, economically meaningful restriction makes multidimensional screening **manageable**.
- Strong **predictions** about optimal mechanisms in applications.

Concluding remarks

Things that I **like** about this paper:

- A fairly natural, economically meaningful restriction makes multidimensional screening **manageable**.
- Strong **predictions** about optimal mechanisms in applications.
- Searchability appears to be a **portable** notion.

Concluding remarks

Things that I **don't like** about this paper:

Concluding remarks

Things that I **don't like** about this paper:

- **Still not easy** to get to closed-form economically insightful solutions (incremental nesting is not the nicest constraint).

Concluding remarks

Things that I **don't like** about this paper:

- **Still not easy** to get to closed-form economically insightful solutions (incremental nesting is not the nicest constraint).
- Not clear **why** the mechanism designer should care about making the life of the agent easier

Concluding remarks

Things that I **don't like** about this paper:

- **Still not easy** to get to closed-form economically insightful solutions (incremental nesting is not the nicest constraint).
- Not clear **why** the mechanism designer should care about making the life of the agent easier (cf. criticism of simplicity in Li ⓡ Dworczak, 2024).

Concluding remarks

Things that I **don't like** about this paper:

- **Still not easy** to get to closed-form economically insightful solutions (incremental nesting is not the nicest constraint).
- Not clear **why** the mechanism designer should care about making the life of the agent easier (cf. criticism of simplicity in Li ⓡ Dworczak, 2024).
- (Grid-)searchability is **too restrictive** in complex non-linear environments.

Concluding remarks

- **Still not easy** to get to closed-form economically insightful solutions (incremental nesting is not the nicest constraint).

  $\Longrightarrow$ **Search for alternative frameworks**

- Not clear **why** the mechanism designer should care about making the life of the agent easier (cf. criticism of simplicity in Li ⓡ Dworczak, 2024).

  $\Longrightarrow$ **Worst-case optimality foundation for searchability**

- (Grid-)searchability is **too restrictive** in complex non-linear environments.

  $\Longrightarrow$ **Other notions of search**

Appendix

A search-game formulation

Definition of search game:

Consider a finite ordered menu $x^1 \prec \cdots \prec x^m$.

- **Ascending search game:** Start at option 1. At each $s < m$, the agent chooses whether to:
  - **clinch** x^s and stop;
  - **continue** to option $s + 1$.
  - at the final node m, agent can choose **any option**.

A search-game formulation

Definition of search game:

Consider a finite ordered menu $x^1 \prec \cdots \prec x^m$.

- **Ascending search game:** Start at option 1. At each $s < m$, the agent chooses whether to:
  - **clinch** x^s and stop;
  - **continue** to option $s + 1$.
  - at the final node m, agent can choose **any option**.

- **Descending search game:** The same procedure in reverse.

Searchability and one-step simplicity

- Pycia and Troyan (2023): A strategy is **one-step dominant** if, at each on-path node, there is a plan for the immediate next node whose **worst outcome** is weakly preferred to the **best outcome from deviating now**.
- A game is **one-step simple** if every type has a one-step dominant strategy.

Searchability and one-step simplicity

- Pycia and Troyan (2023): A strategy is **one-step dominant** if, at each on-path node, there is a plan for the immediate next node whose **worst outcome** is weakly preferred to the **best outcome from deviating now**.
- A game is **one-step simple** if every type has a one-step dominant strategy.

Proposition

*In a regular environment, a finite irredundant menu is **searchable** if and only if both its ascending and descending **search games are one-step simple**.*

Searchability and one-step simplicity

- Pycia and Troyan (2023): A strategy is **one-step dominant** if, at each on-path node, there is a plan for the immediate next node whose **worst outcome** is weakly preferred to the **best outcome from deviating now**.
- A game is **one-step simple** if every type has a one-step dominant strategy.

Proposition

*In a regular environment, a finite irredundant menu is **searchable** if and only if both its ascending and descending **search games are one-step simple**.*

Intuition: Comparing clinching now with continuing and clinching the next option is enough to find an optimum.

Back