Computing Walrasian Equilibrium

Renato Paes Leme
(Google)

Sam Wong
(Berkeley)

supplies:
flour,
milk,
vegetables,
medicine,
paper,
...

demand:
bakeries,
hospitals,
households,
schools,
...

supplies:

flour,

milk,

vegetables,

medicine,

paper,

...

Task: Allocate supplies efficiently to satisfy the demands of the city.

demand:

bakeries,

hospitals,

households,

schools,

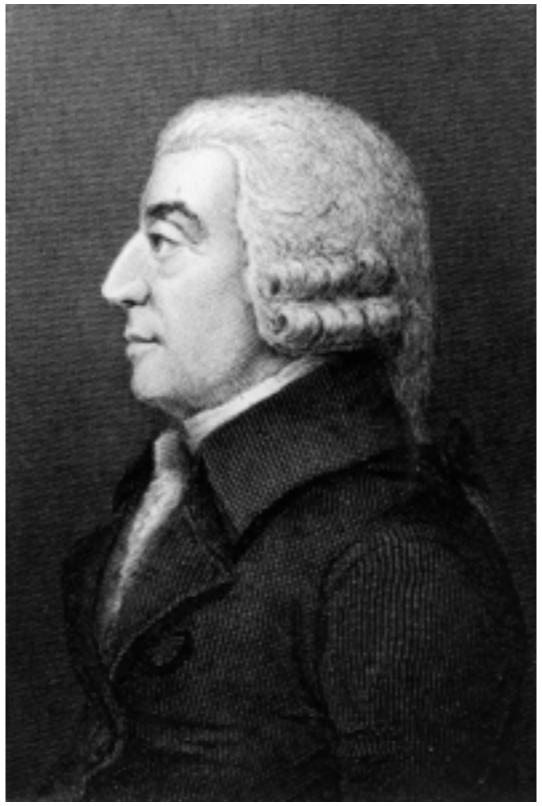
...

supplies:
flour,
milk,
vegetables,
medicine,
paper,
...

demand:
bakeries,
hospitals,
households,
schools,
...

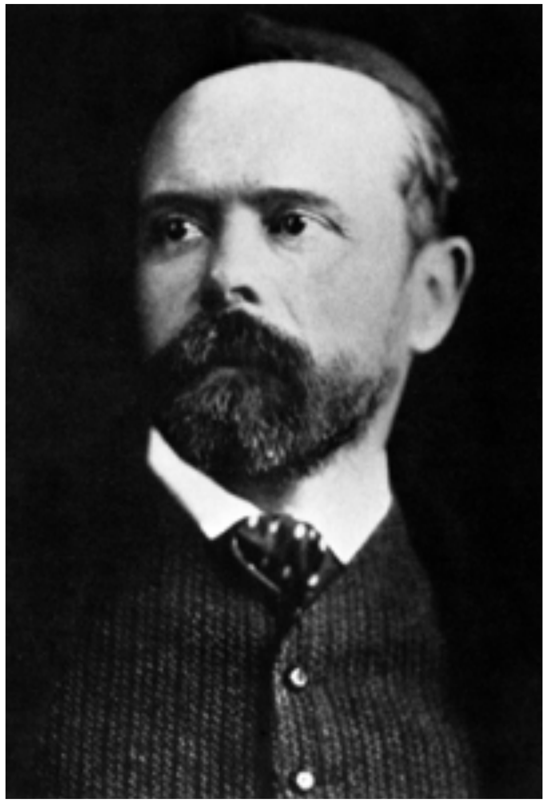
Invisible Hand
of the market

Theory of Market Equilibrium



- Adam Smith: "Wealth of the Nations" (1776): invisible hand

- Leon Walras: "Elements of Pure Economics" (1874): mathematical theory of market equilibrium

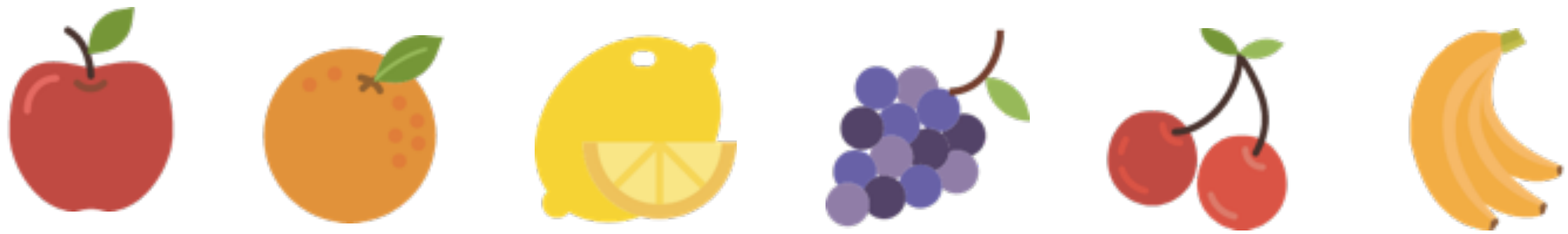


- Arrow-Debreu (1950's): general equilibrium theory

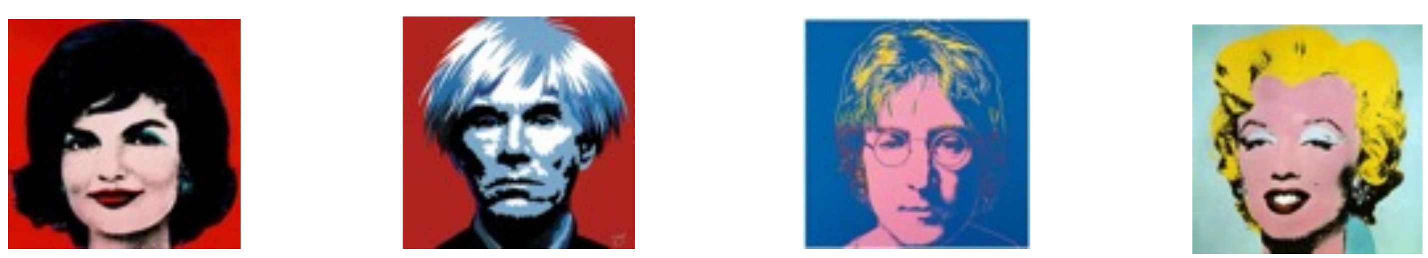
- Kelso-Crawford (1982): discrete and combinatorial theory of market equilibr.

Market equilibrium

n goods

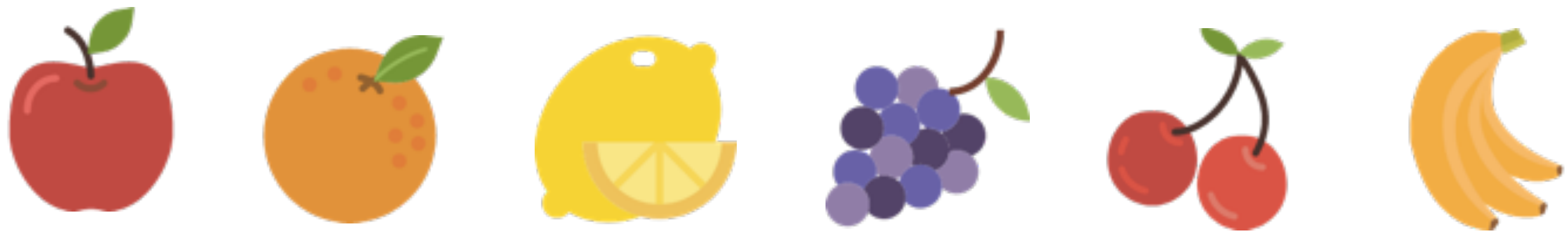


m buyers

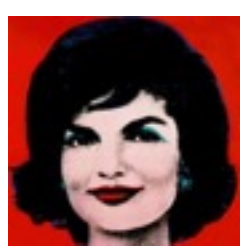


Market equilibrium

n goods



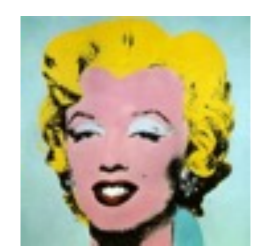
m buyers



v_1

v_2

v_3



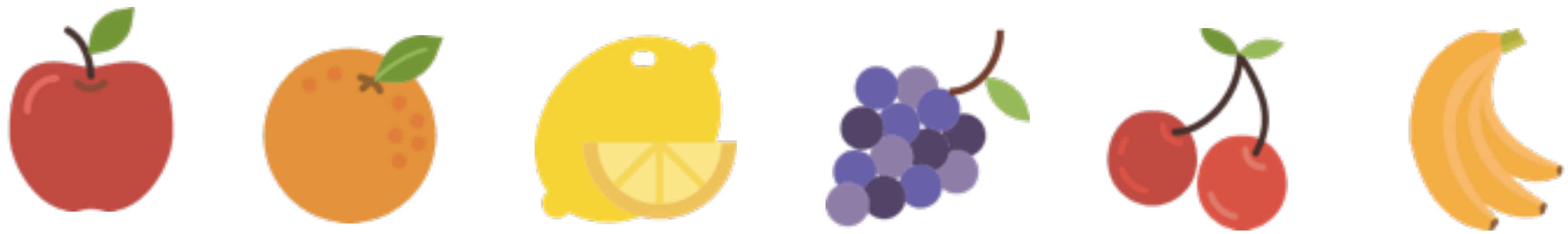
v_4

- Valuations $v_i : 2^N \to \mathbb{R}$

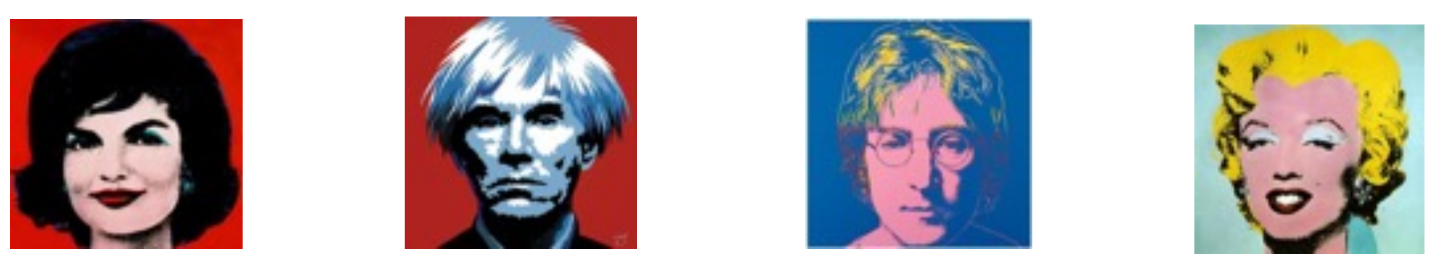
Market equilibrium

p_1 p_2 p_3 p_4 p_5 p_6

n goods



m buyers



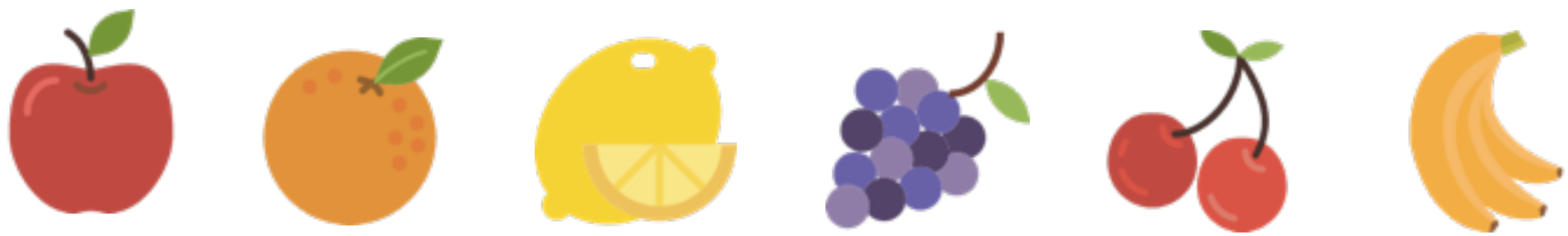
v_1 v_2 v_3 v_4

- Valuations $v_i : 2^N \to \mathbb{R}$

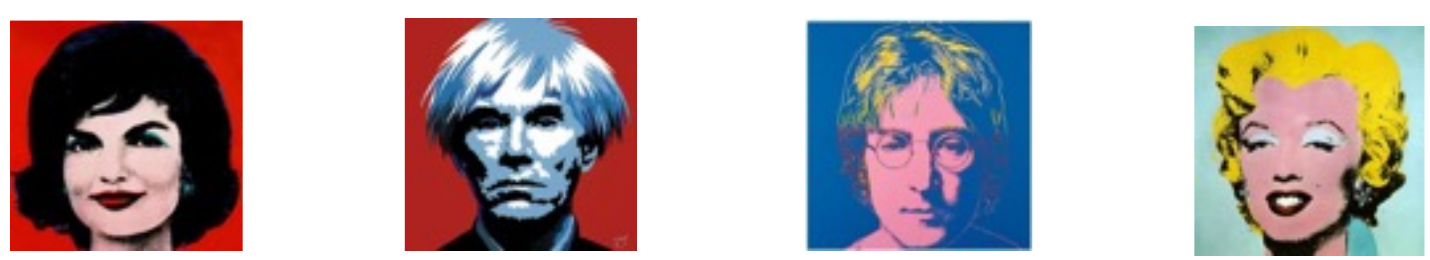
Market equilibrium

p_1 p_2 p_3 p_4 p_5 p_6

n goods



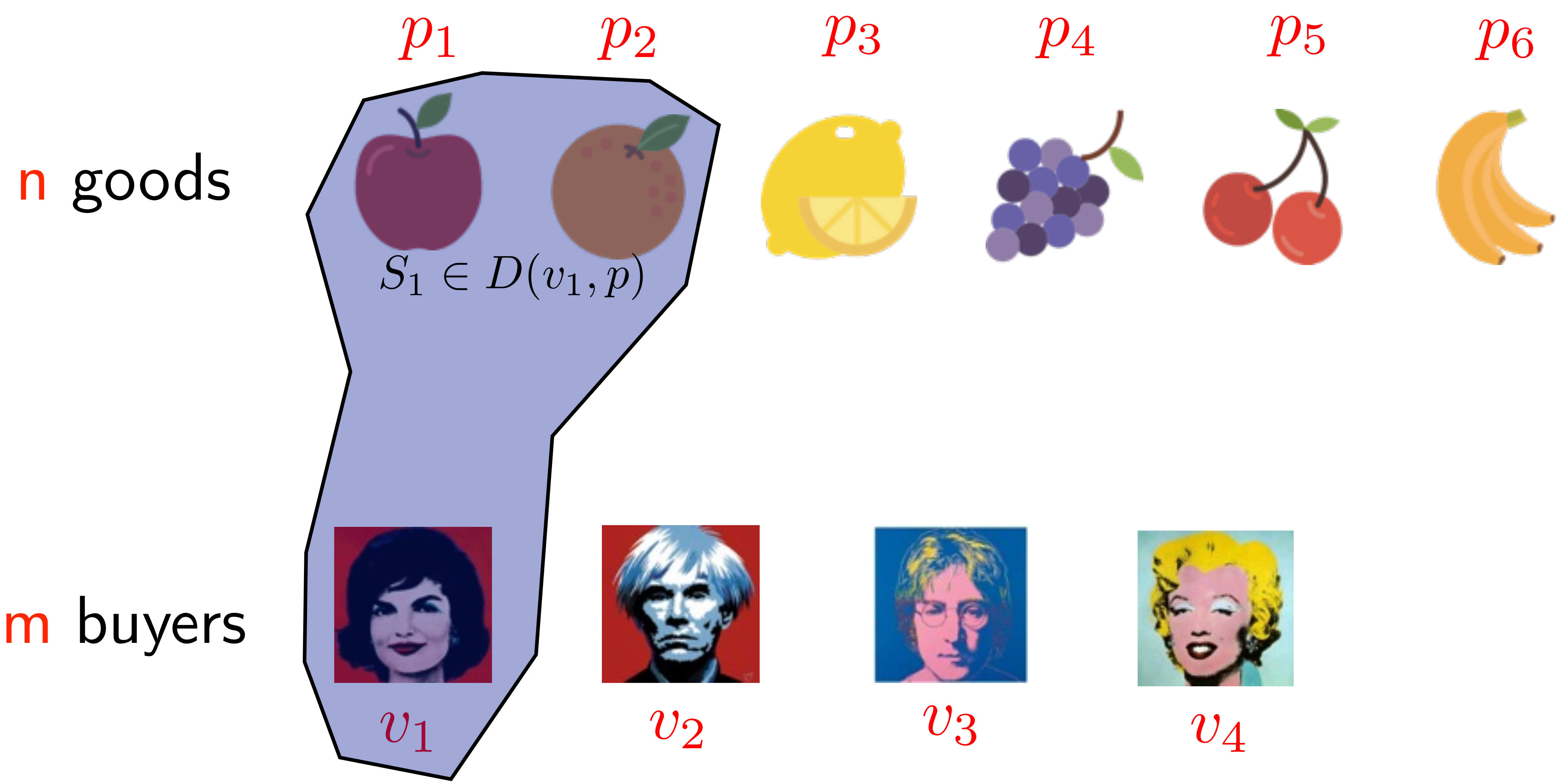
m buyers



v_1 v_2 v_3 v_4

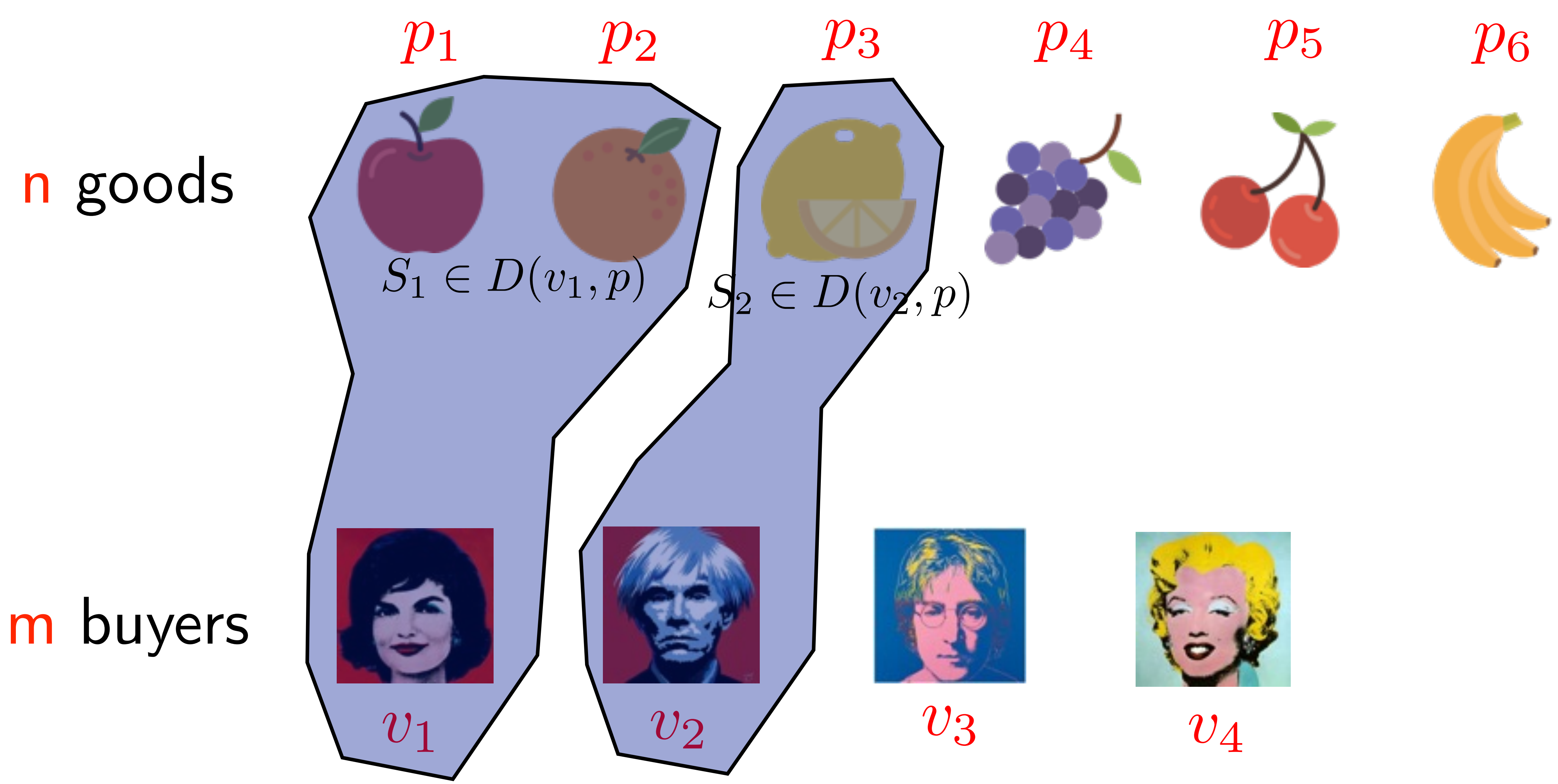
- Valuations $v_i : 2^N \to \mathbb{R}$
- Demands $D(v_i, p) = \text{argmax}_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i]$

Market equilibrium



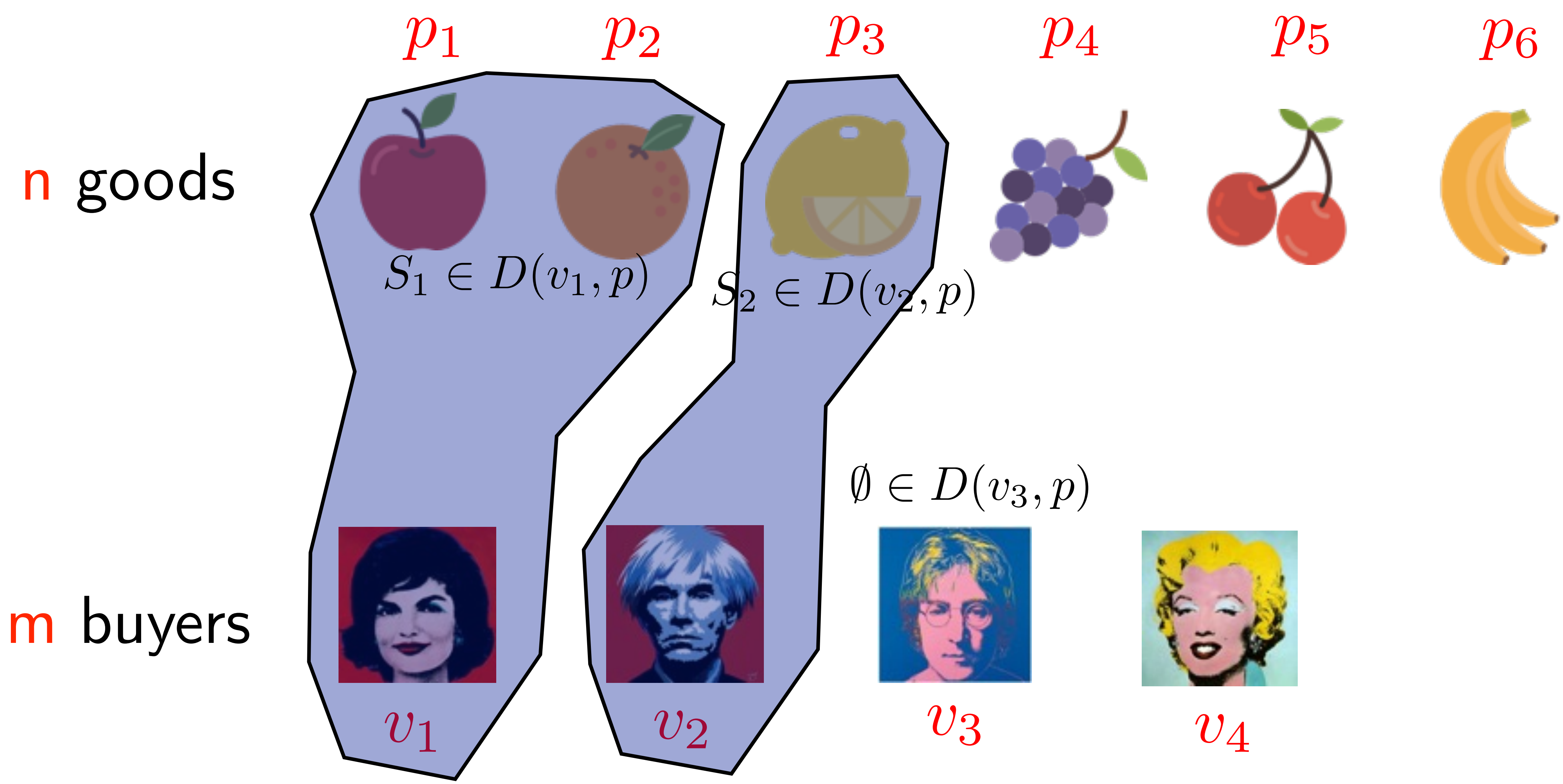
- Valuations $v_i : 2^N \to \mathbb{R}$
- Demands $D(v_i, p) = \text{argmax}_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i]$

Market equilibrium



- Valuations $v_i : 2^N \to \mathbb{R}$
- Demands $D(v_i, p) = \text{argmax}_{S \subseteq N}[v_i(S) - \sum_{i \in S} p_i]$

Market equilibrium

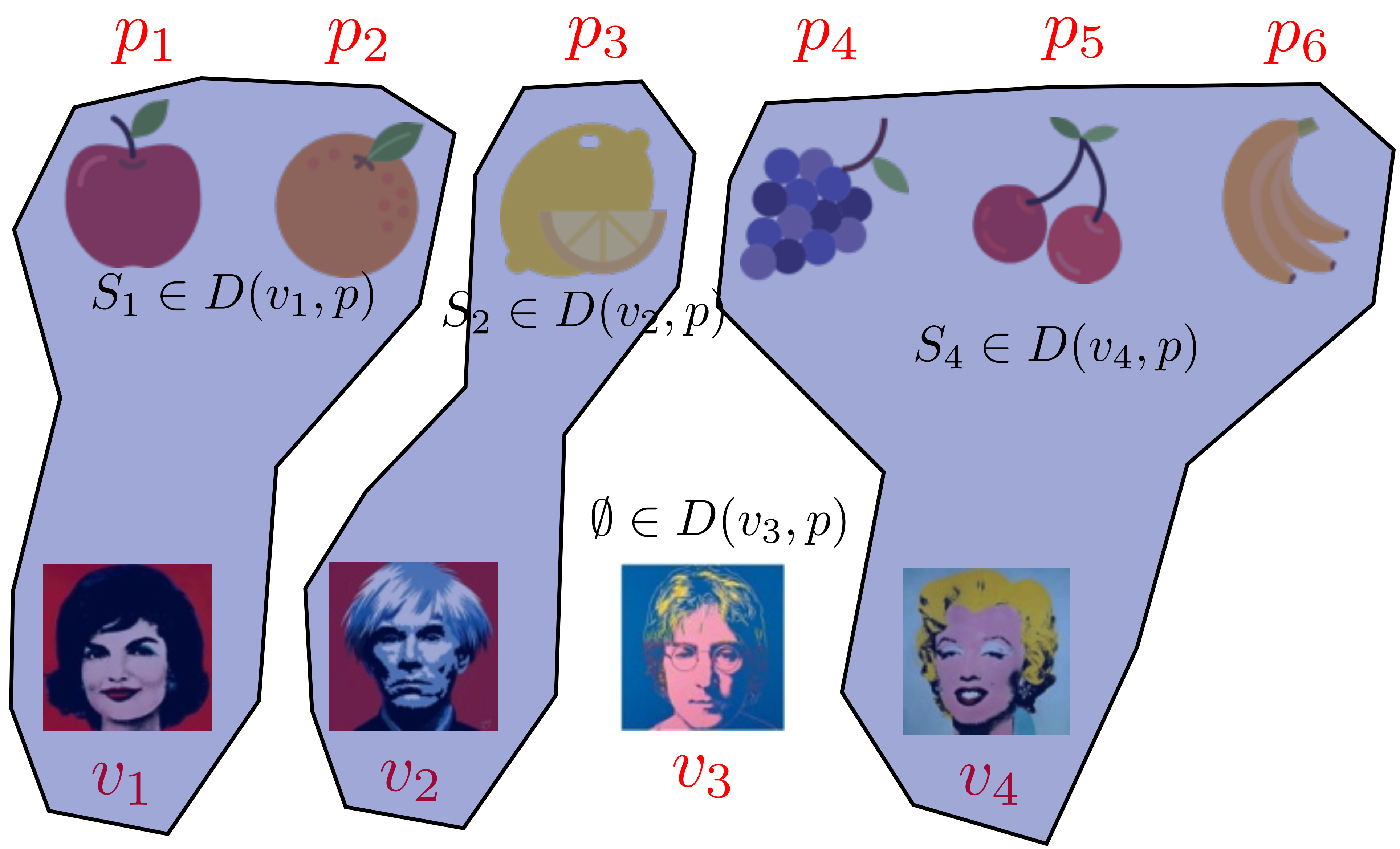


- Valuations $v_i : 2^N \to \mathbb{R}$
- Demands $D(v_i, p) = \text{argmax}_{S \subseteq N}[v_i(S) - \sum_{i \in S} p_i]$

Market equilibrium

n goods

m buyers



- Valuations $v_i : 2^N \to \mathbb{R}$
- Demands $D(v_i, p) = \text{argmax}_{S \subseteq N} [v_i(S) - \sum_{i \in S} p_i]$

Market equilibrium

- Market equilibrium: prices $p \in \mathbb{R}^n$ s.t. $S_i \in D(v_i, p)$ i.e. each good is demanded by **exactly** one buyer.

First Welfare Theorem: in equilibrium the welfare $\sum_i v_i(S_i)$ is maximized.

(proof: LP duality)

How do markets converge to equilibrium prices ?

How to compute a Walrasian equilibrium ?

How to access the input

Microscopic

Macroscopic

Telescopic

How to access the input

Microscopic

Macroscopic

Telescopic

Value oracle:

given i and S:

query $v_i(S)$.

How to access the input

Microscopic

Value oracle:

given i and S:

query $v_i(S)$.

Macroscopic

Demand oracle:

given i and p:

query $S \in D(v_i, p)$

Telescopic

How to access the input

Microscopic

Value oracle:

given i and S:

query $v_i(S)$.

Macroscopic

Demand oracle:

given i and p:

query $S \in D(v_i, p)$

Telescopic

Aggregate Demand:

given p, query.

$\sum_i S_i; S_i \in D(v_i, p)$

Algorithms for computing equilibria (general case)

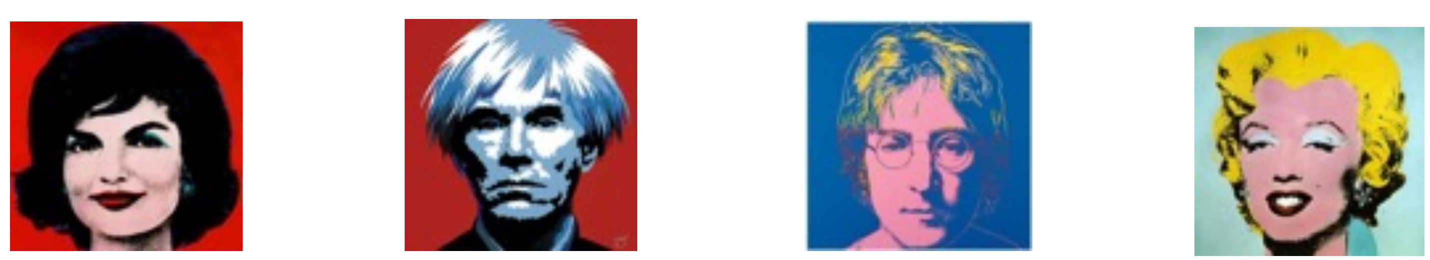
Algorithm	Oracle Access	Running time
tatonnement (trial-and-error) [Walras, Kelso-Crawford, ...]		

Walrasian tatonnement

p_1 p_2 p_3 p_4 p_5 p_6

n goods

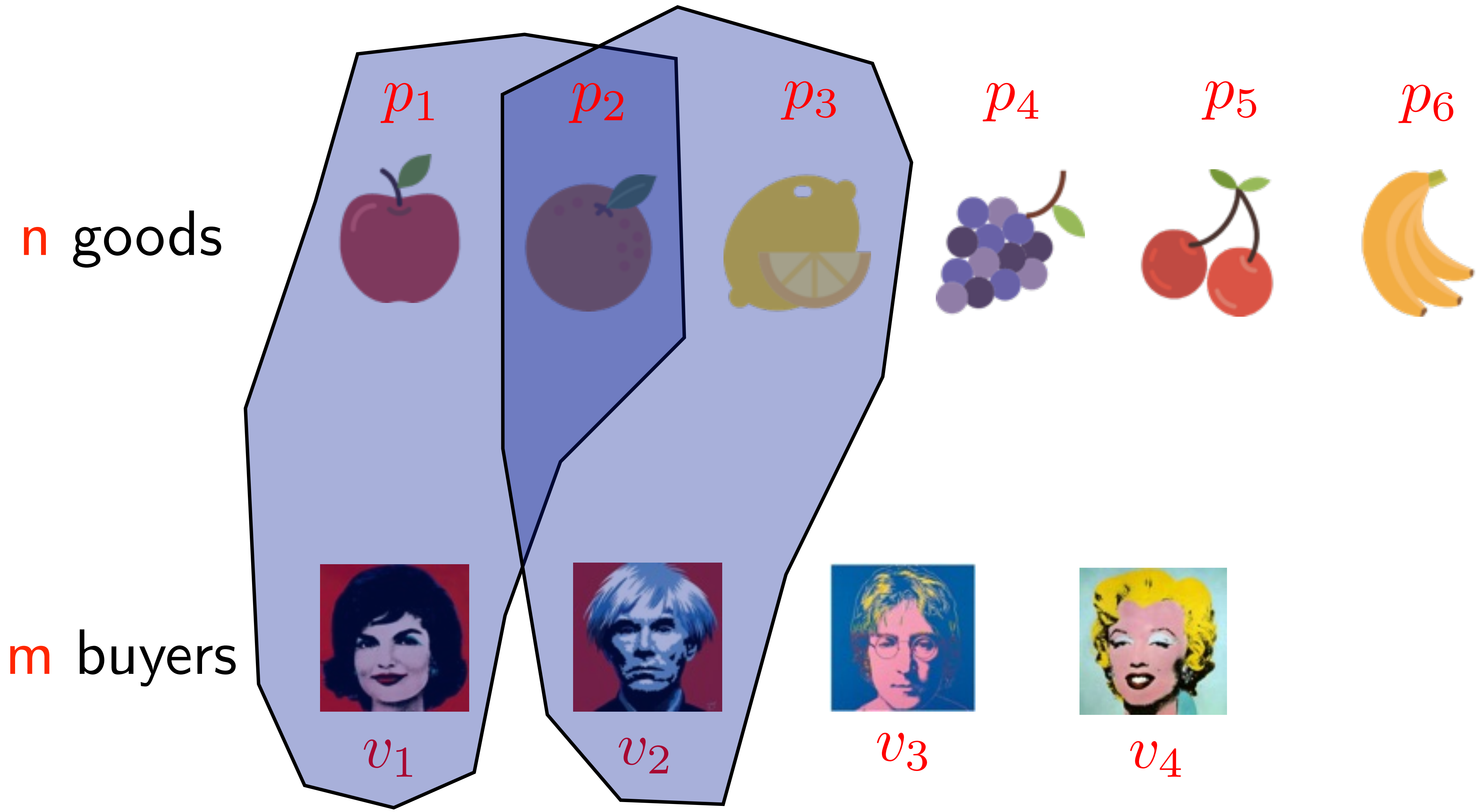
m buyers



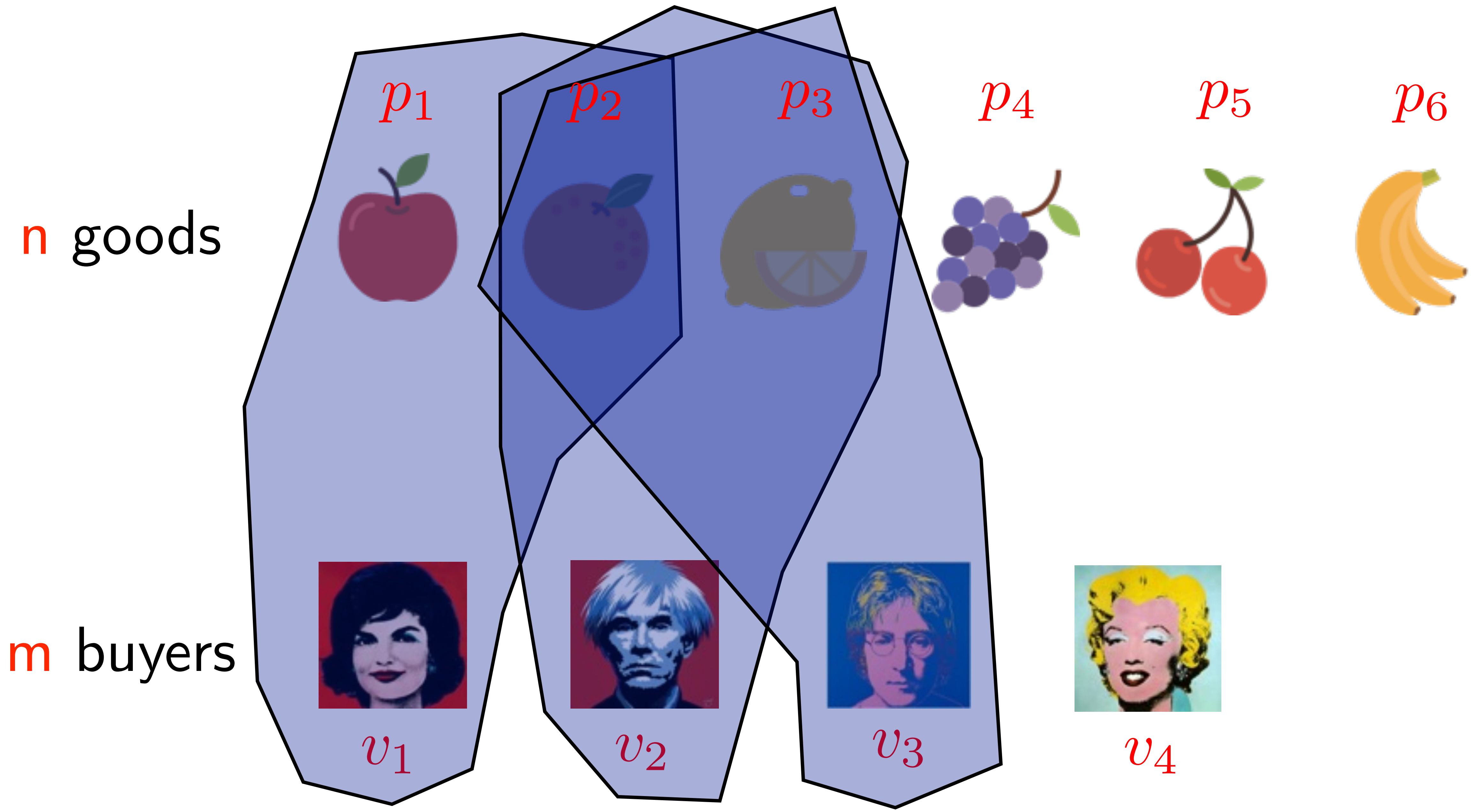
v_1 v_2 v_3 v_4

Walrasian tatonnement

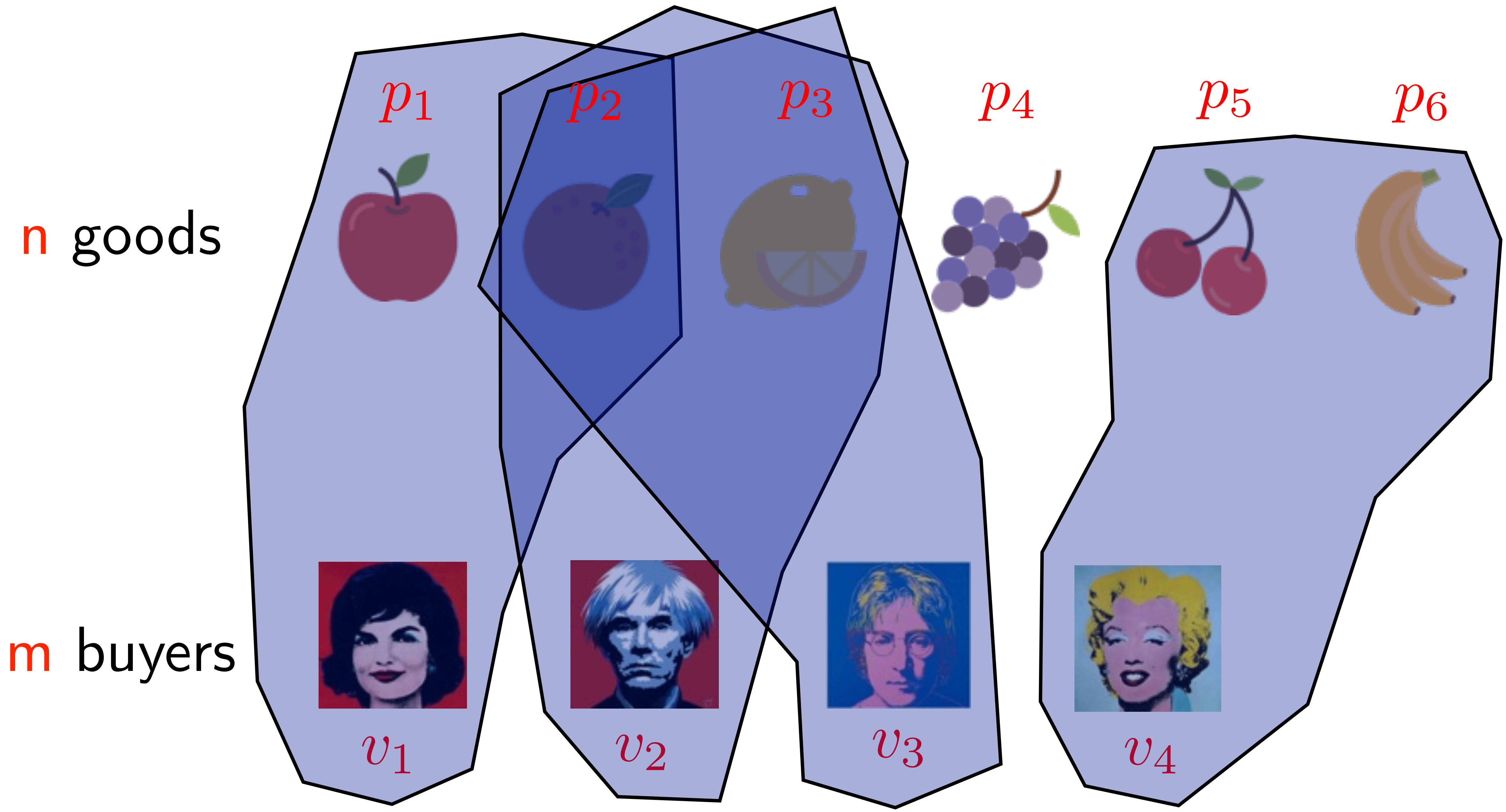
Walrasian tatonnement



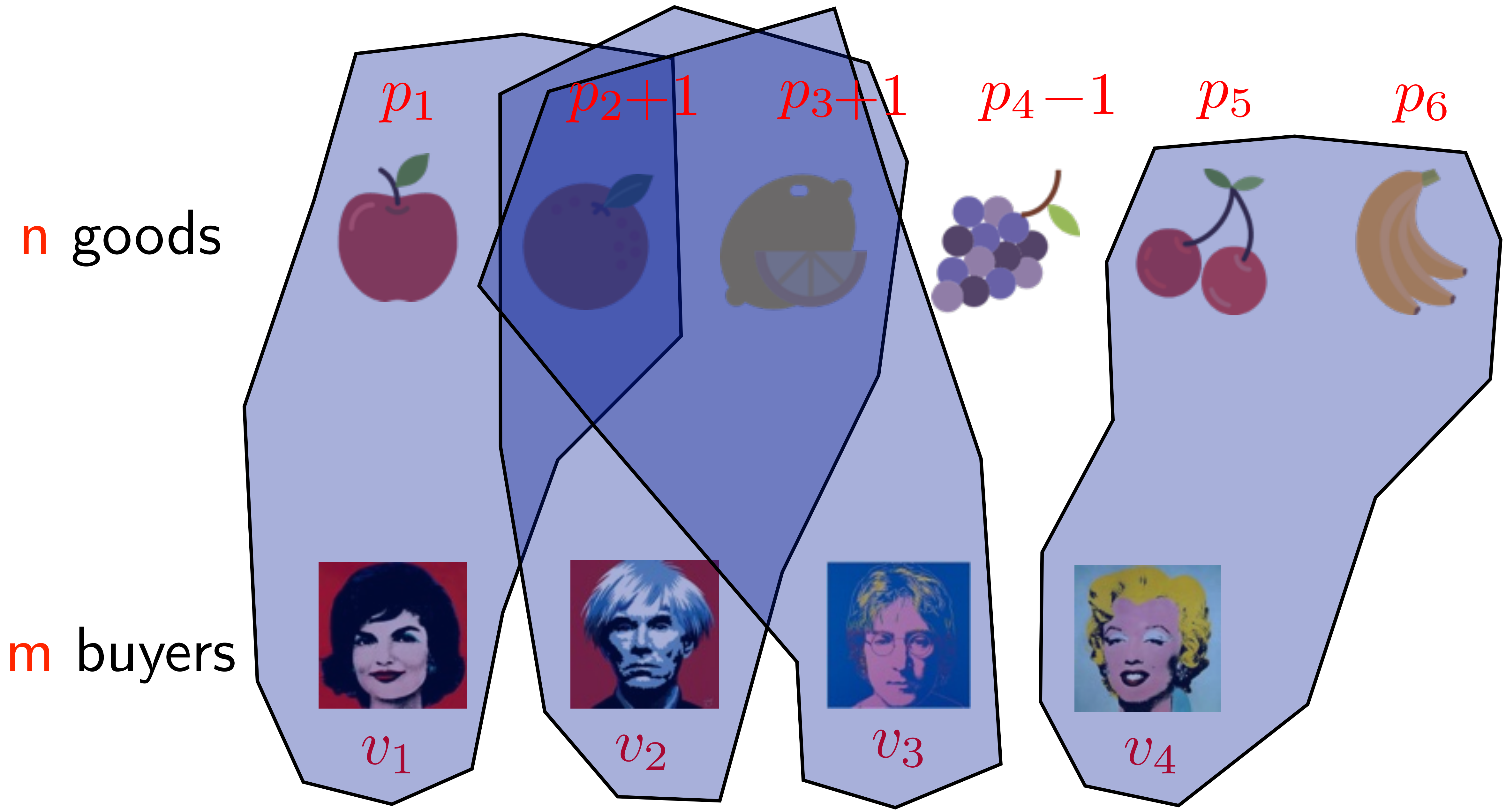
Walrasian tatonnement



Walrasian tatonnement



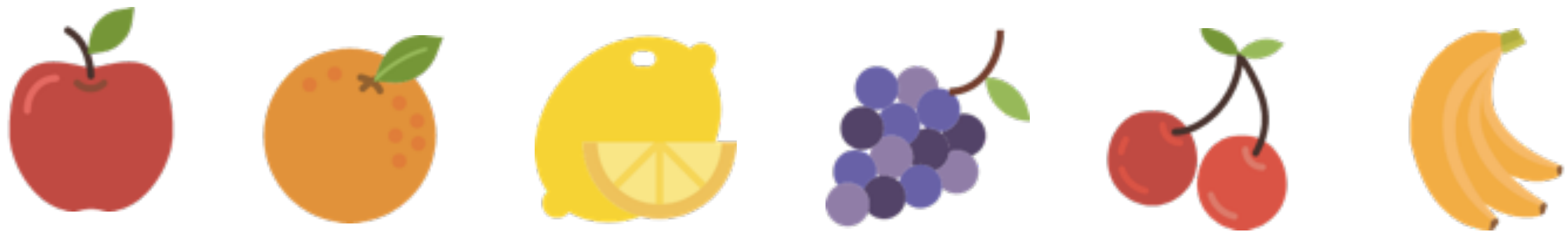
Walrasian tatonnement



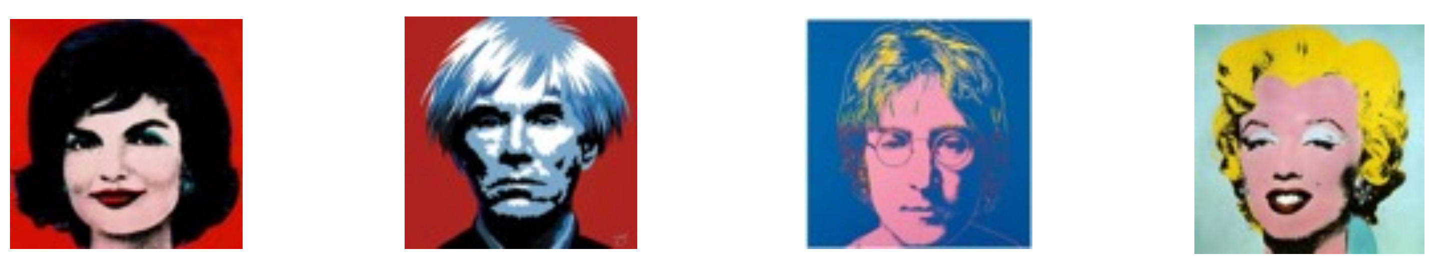
Walrasian tatonnement

n goods

p_1 $p_2{+}1$ $p_3{+}1$ p_4-1 p_5 p_6



m buyers



v_1 v_2 v_3 v_4

Walrasian tatonnement

p_1 $p_2{+}1$ $p_3{+}1$ p_4-1 p_5 p_6

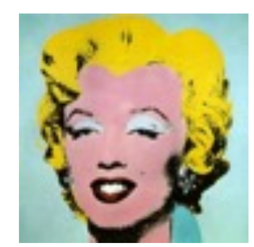
n goods





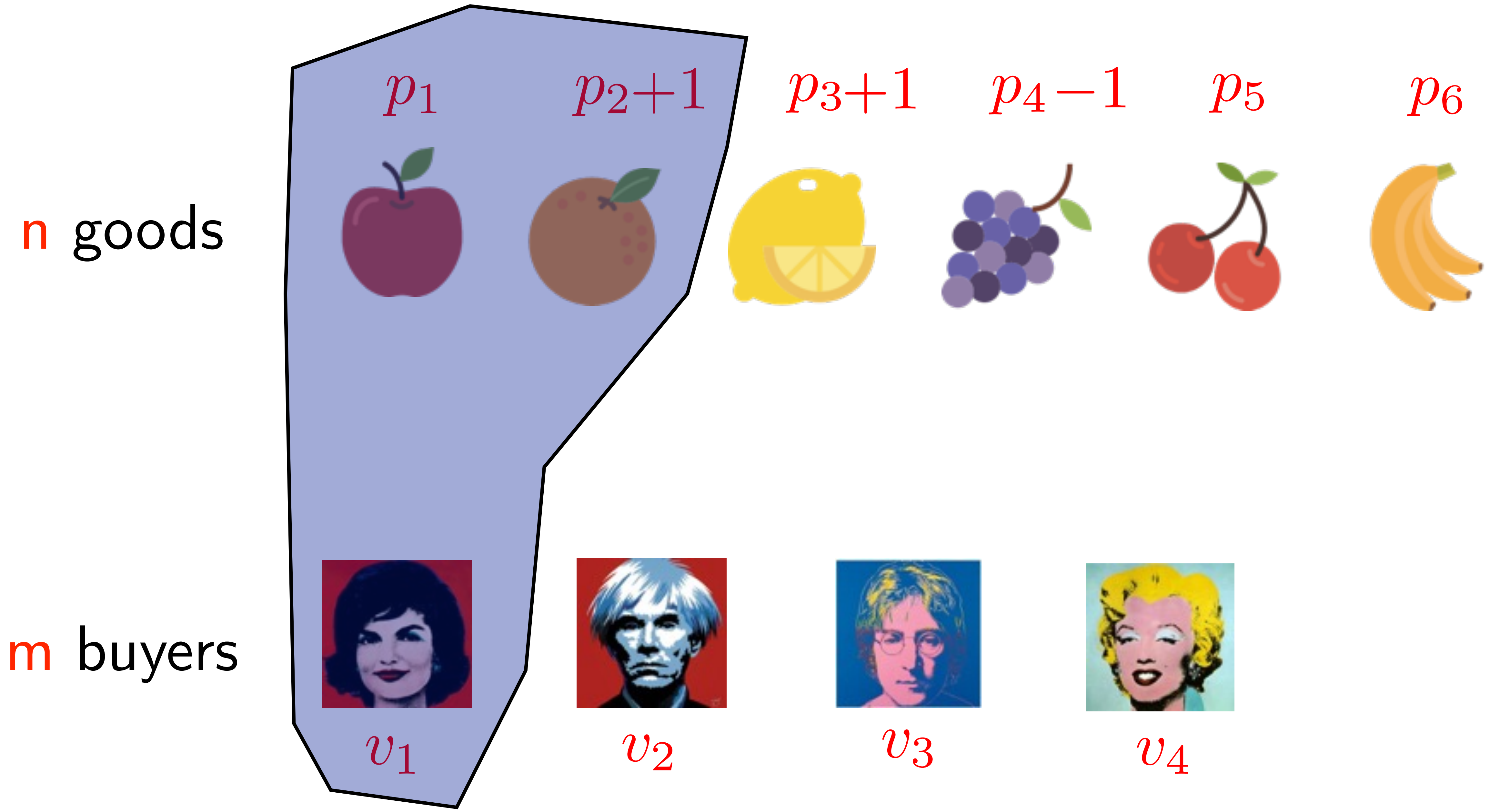

m buyers



v_1 v_2 v_3 v_4

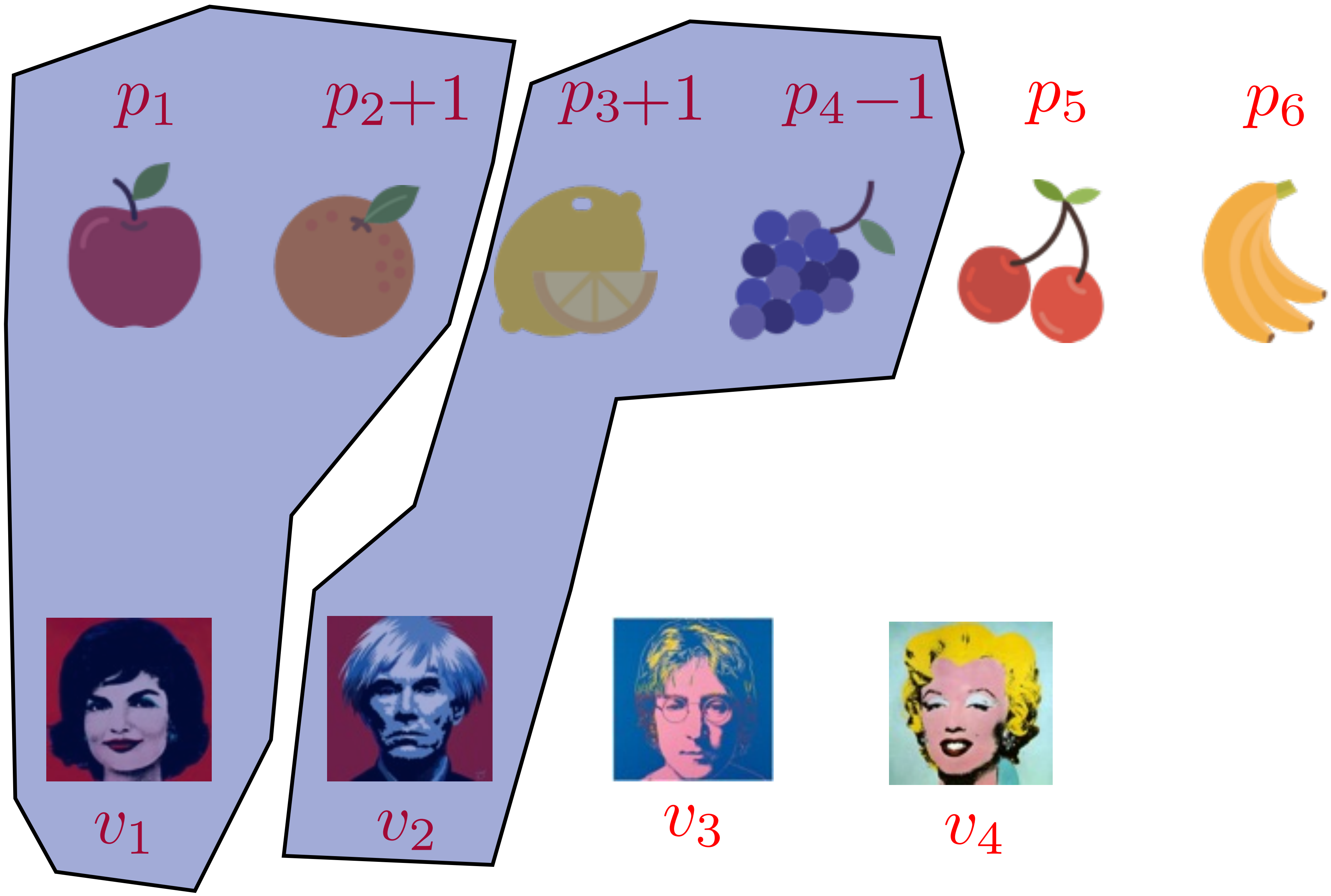
Walrasian tatonnement



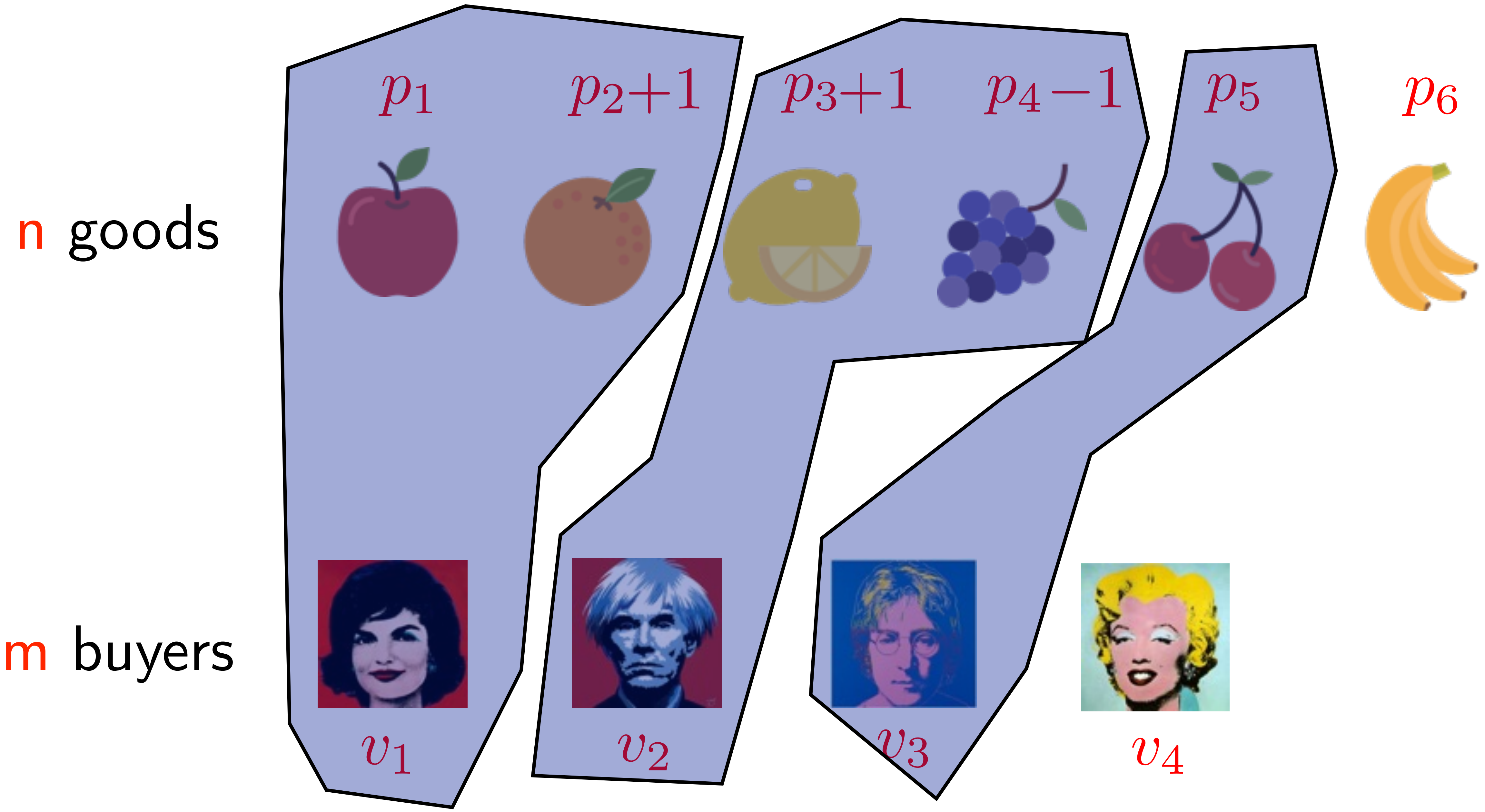
Walrasian tatonnement

n goods

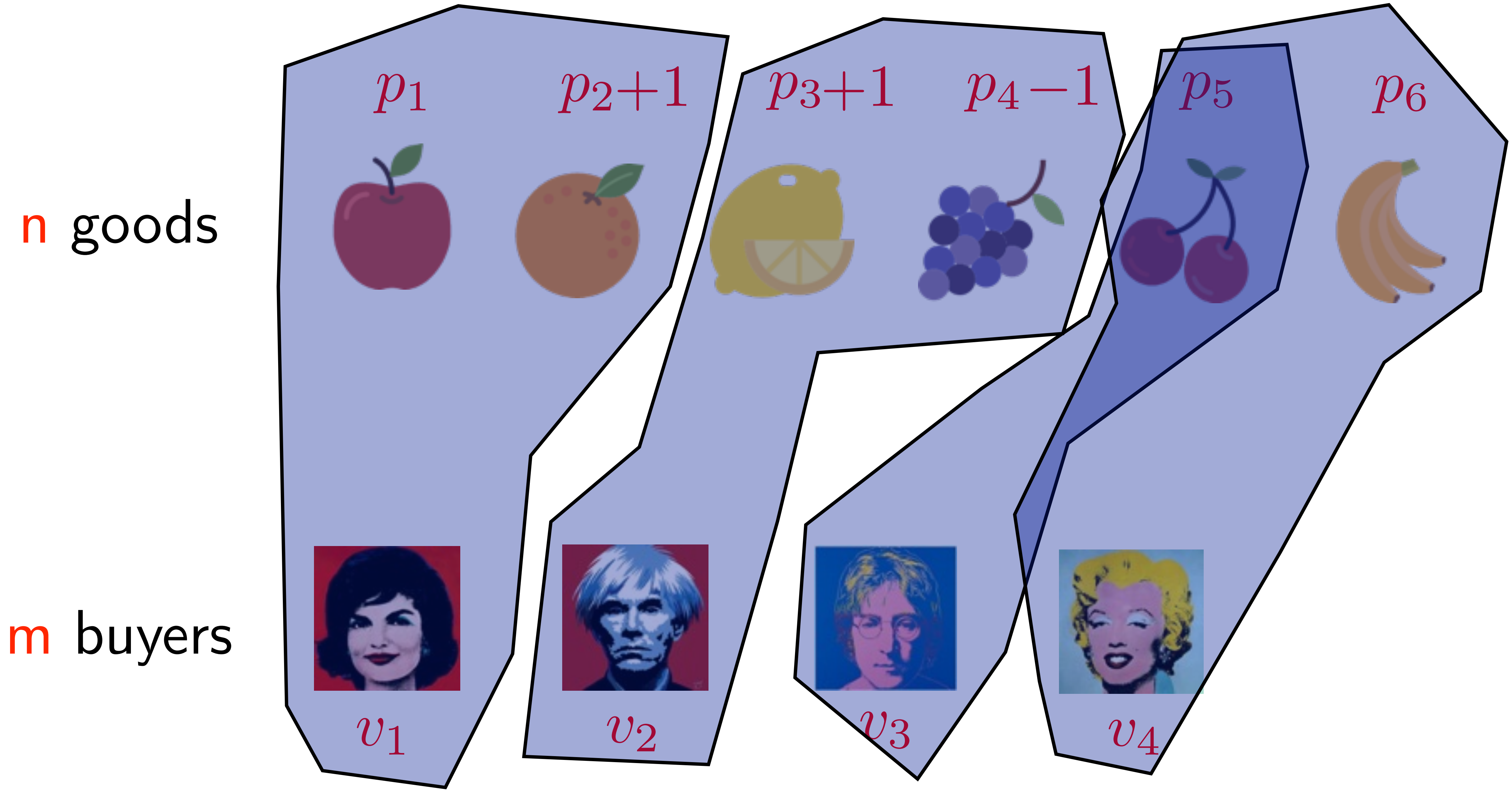
m buyers



Walrasian tatonnement



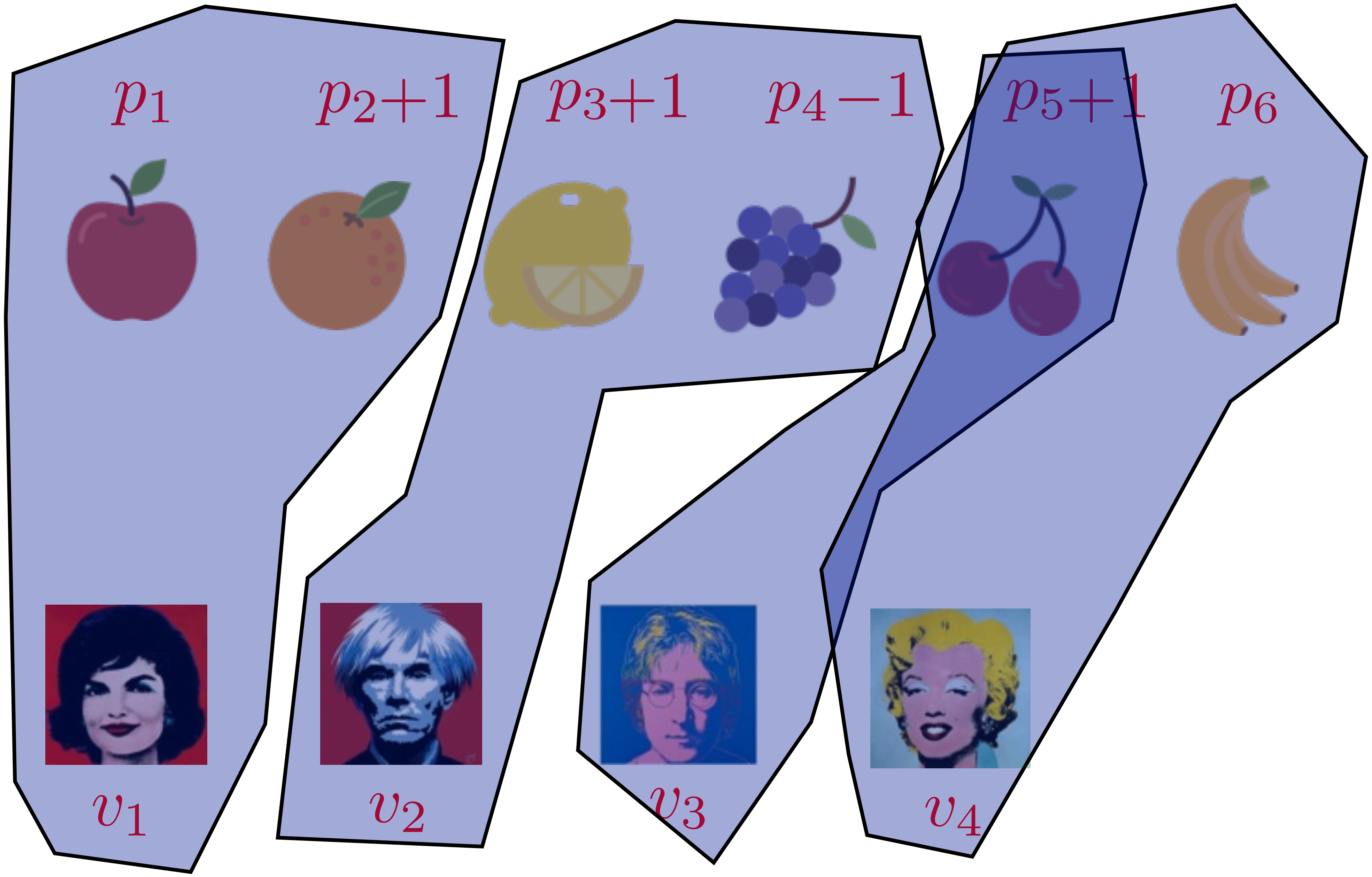
Walrasian tatonnement



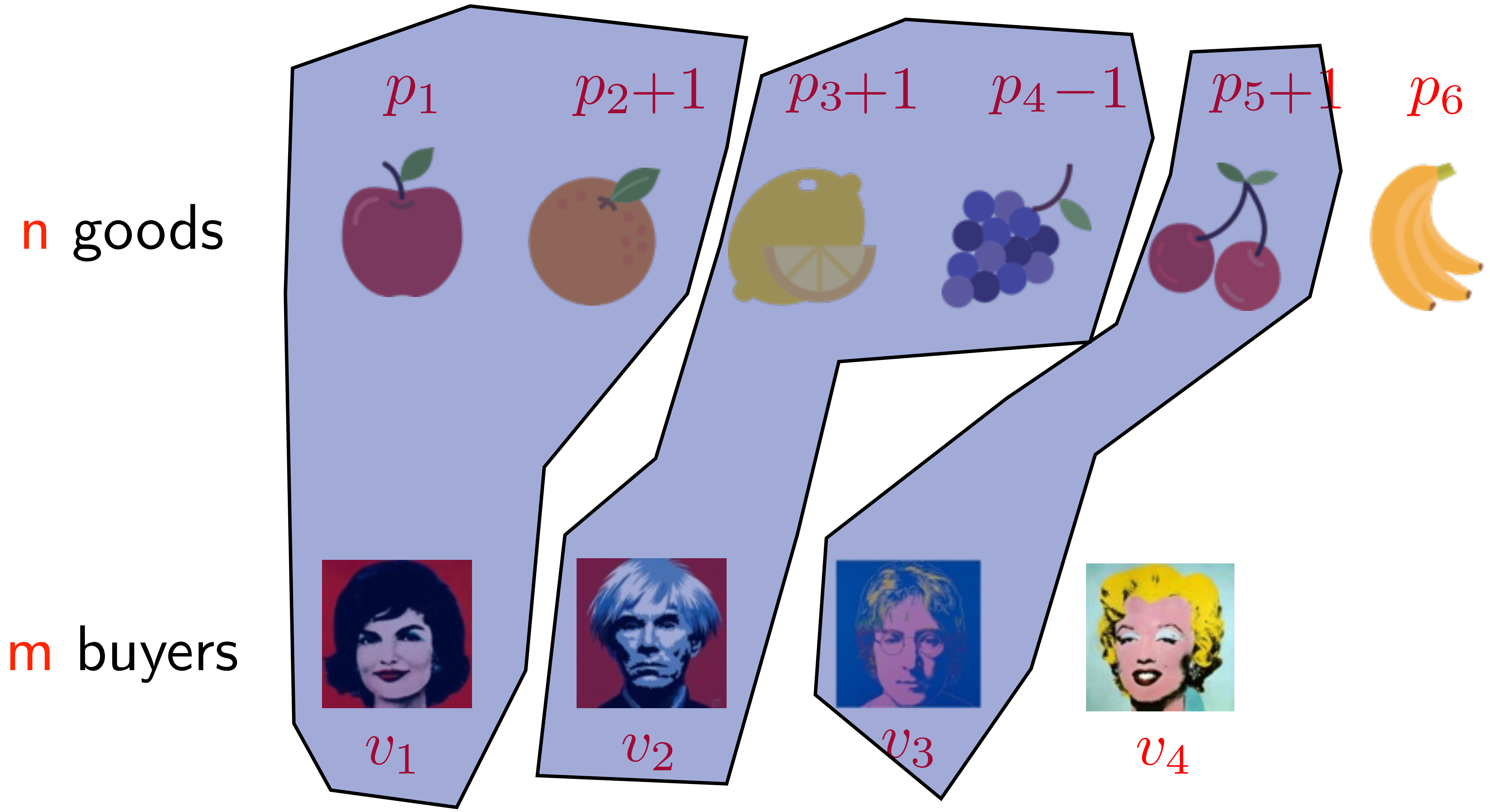
Walrasian tatonnement

n goods

m buyers



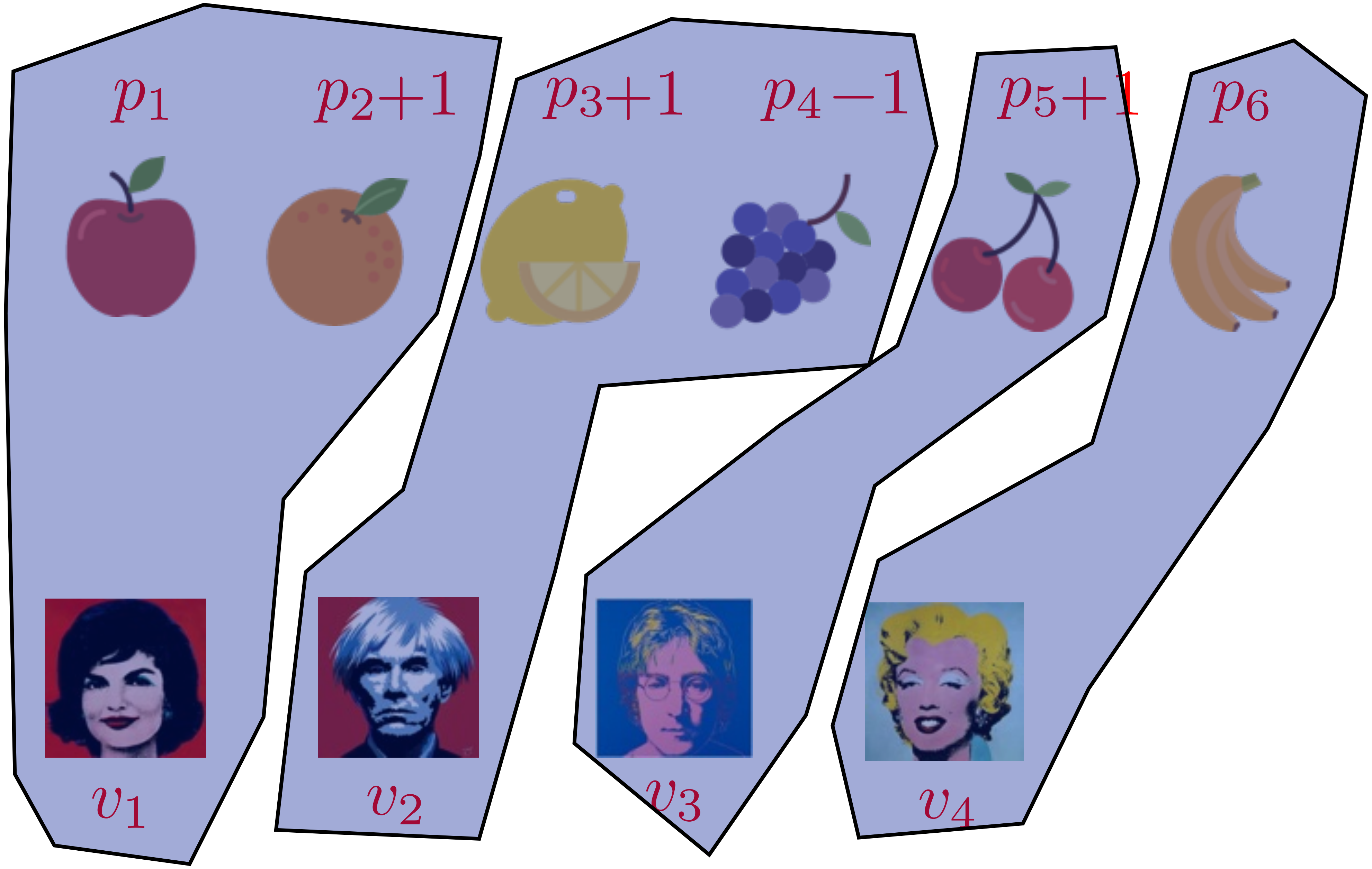
Walrasian tatonnement



Walrasian tatonnement

n goods

m buyers



Gradient Descent Interpretation

- [Kelso-Crawford] analyzes it and shows convergence under a condition called gross substitutes.
  - pseudo poly algorithm

Gradient Descent Interpretation

- [Kelso-Crawford] analyzes it and shows convergence under a condition called gross substitutes.
  - pseudo poly algorithm

- [Ausubel] defined the potential:

$$f(p) = \sum_i \max_S [v_i(S) - p(S)] + p([n])$$

  such that gradient descent is exactly tatonnement:

$$\partial_j f(p) = 1 - [\text{total demand for } j]$$

- If equilibrium exists then equil prices = $\operatorname{argmin} f(p)$

Algorithms for computing equilibria (general case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly

Algorithms for computing equilibria (general case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly
Linear programming [Nisan-Segal]	demand + value oracle	poly time

Algorithms for computing equilibria (general case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly
Linear programming [Nisan-Segal]	demand + value oracle	poly time
this paper	aggregate demand	poly time $\tilde{O}(n^2 \cdot T_{AD} + n^5)$

From LP to convex optimization

- Nisan and Segal LP:

$$\min \sum_i u_i + p([n])$$

$$u_i \geq v_i(S) - p(S), \forall i, S$$

From LP to convex optimization

- Nisan and Segal LP:

  $$\min \sum_i u_i + p([n])$$

  $$u_i \geq v_i(S) - p(S), \forall i, S$$

- demand oracle finds separating constraint
- value oracle to add the hyperplane

From LP to convex optimization

- Nisan and Segal LP:

  $$\min \sum_i u_i + p([n])$$

  $$u_i \geq v_i(S) - p(S), \forall i, S$$

- demand oracle finds separating constraint
- value oracle to add the hyperplane

- Idea: using cutting plane method to minimize

  $$f(p) = \sum_i [\max_S v_i(S) - p(S)] + p([n])$$

From LP to convex optimization

- Nisan and Segal LP:

  $\min \sum_i u_i + p([n])$

  $u_i \geq v_i(S) - p(S), \forall i, S$

- demand oracle finds separating constraint
- value oracle to add the hyperplane

- Idea: using cutting plane method to minimize

  $$f(p) = \sum_i [\max_S v_i(S) - p(S)] + p([n])$$

- Two issues with black box application:
  - **Evaluate f**: ellipsoid and cutting plane need $f(p), \partial f(p)$
  - **Approximation**: give only approximate solutions

From LP to convex optimization

- **Optimizing only using the gradient**
  We adapt the cutting plane algorithm of Lee-Sidford-Wong'15 to optimize f using only $\partial f(p)$

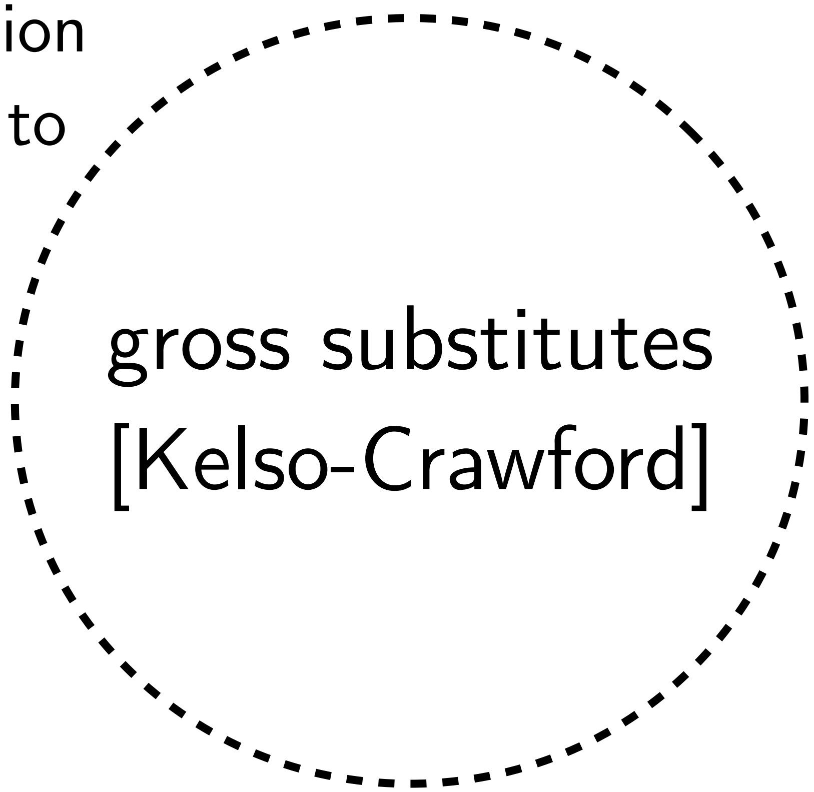
- **Obtaining exact solutions**
  - Exact solution is only known for LPs [Khachiyan]
  - idea: explore the connection of this program and LP
  - But we have restricted access to constraints (only via aggregate demand oracle)
  - Only a restricted perturbation is enough.

Gross substitutes case

Gross substitutes case

necessary and "sufficient" condition for tatonnement to converge

"increase in the price for one good doesn't decrease demand for other good."



Gross substitutes case

necessary and "sufficient" condition for tatonnement to converge

"increase in the price for one good doesn't decrease demand for other good."

gross substitutes [Kelso-Crawford]

valuated matroids [Dress-Wenzel]

generalization of Grassman-Plucker relations, when can $v(S) - \sum_S p_j$ be optimized using Greedy algo

Gross substitutes case

necessary and "sufficient" condition for tatonnement to converge

"increase in the price for one good doesn't decrease demand for other good."

gross substitutes [Kelso-Crawford]

valuated matroids [Dress-Wenzel]

generalization of Grassman-Plucker relations, when can $v(S) - \sum_S p_j$ be optimized using Greedy algo

(if $v(S) \in \{0, -\infty\}$ those are matroids).

Gross substitutes case

necessary and "sufficient" condition for tatonnement to converge

"increase in the price for one good doesn't decrease demand for other good."

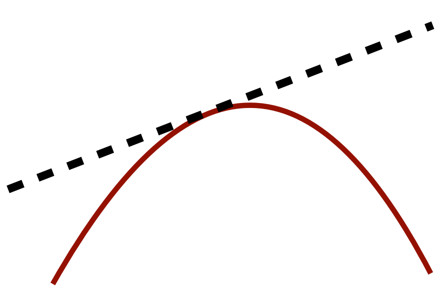
gross substitutes [Kelso-Crawford]

valuated matroids [Dress-Wenzel]

generalization of Grassman-Plucker relations, when can $v(S) - \sum_S p_j$ be optimized using Greedy algo

(if $v(S) \in \{0, -\infty\}$ those are matroids).

discrete concavity [Murota-Shioura]

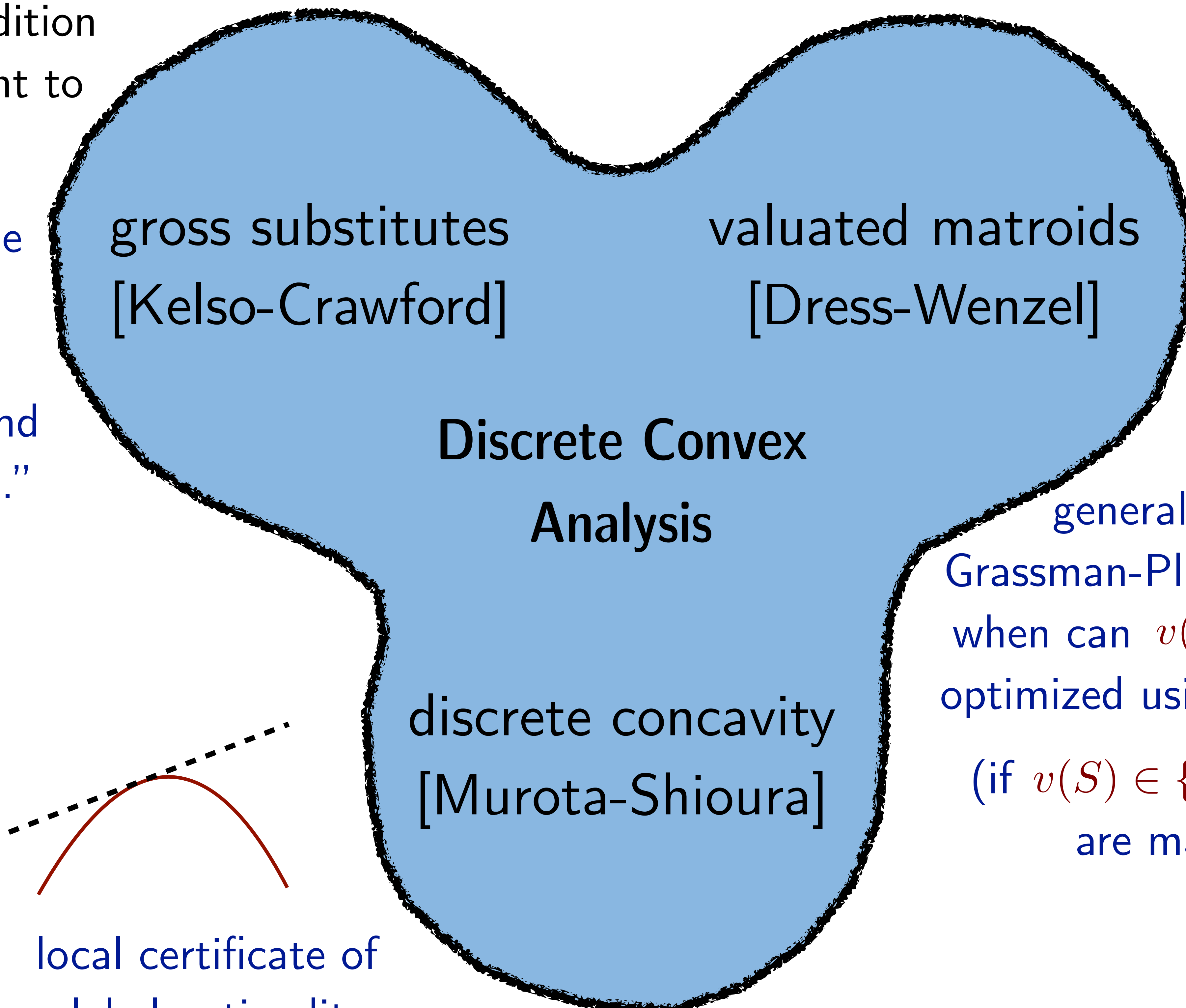


local certificate of global optimality

Gross substitutes case

necessary and "sufficient" condition for tatonnement to converge

"increase in the price for one good doesn't decrease demand for other good."



generalization of Grassman-Plucker relations, when can $v(S) - \sum_S p_j$ be optimized using Greedy algo

(if $v(S) \in \{0, -\infty\}$ those are matroids).

local certificate of global optimality

Algorithms for computing equilibria (gross substitutes case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly

Algorithms for computing equilibria (gross substitutes case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly
Combinatorial flow-based algos [Murota]	value oracle	strong poly time $\tilde{O}(mn^3 \cdot T_V)$

Algorithms for computing equilibria (gross substitutes case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly
Combinatorial flow-based algos [Murota]	value oracle	strong poly time $\tilde{O}(mn^3 \cdot T_V)$
this paper	aggregate demand	$\tilde{O}(n \cdot T_{AD} + n^3)$

Algorithms for computing equilibria (gross substitutes case)

Algorithm	Oracle Access	Running time
tatonnement / gradient descent [Walras, Kelso-Crawford, ...]	aggregate demand	pseudo poly
Combinatorial flow-based algos [Murota]	value oracle	strong poly time $\tilde{O}(mn^3 \cdot T_V)$
this paper	aggregate demand	$\tilde{O}(n \cdot T_{AD} + n^3)$
this paper	value oracle	$\tilde{O}((mn + n^3) \cdot T_V)$

Improving the algorithm for gross substitutes

- Better rounding using structure of gross substitutes gets us to $\tilde{O}(n \cdot T_{AD} + n^3)$
  - plugging $T_{AD} = O(mn^2 \cdot T_V)$ we get $\tilde{O}(mn^3 \cdot T_V)$

- Regularization: gradients are expensive to compute.
  - it takes $O(n^2 \cdot T_V)$ to run Greedy for each buyer.
  - gradients are cheap near the optimal
  - re-use computation from one step to the next
  - we only need precise gradients near the optimum

$$\hat{f}(p) = \sum_i [\max_S v_i(S) - p(S) + \epsilon|S|] + p([n]) - \epsilon n$$

Improving the algorithm for gross substitutes

- Regularized objective:

$$\hat{f}(p) = \sum_i [\max_S v_i(S) - p(S) + \epsilon|S|] + p([n]) - \epsilon n$$

- Same optimal value
- Very accurate near the optimal value, directionally correct for other values.
- Takes only $O(n^2)$ time to compute with $O(mn)$ pre-processing.

Conclusion

- Market equilibrium can be computed:
  - only very aggregated information
  - in $\tilde{O}(n)$ calls to this oracle.

- Questions to think about:
  - Markets that change over time ? New items, new buyers, ... How to update market equilibrium.
  - Strongly poly time algorithms.