Implementing Random Assignments: A Generalization of Birkhoff-von Neumann Theorem

Eric Budish, Yeon-Koo Che, Fuhito Kojima, Paul Milgrom[1]

June 17, 2009

[1]Harvard University, Columbia University, Yale University, Stanford University

Overview

- Social planner wants to assign indivisible objects to agents.
- Examples:
  - School choice problems (NYC, Boston).
  - House allocation in colleges.
  - Course allocation.
- Typical constraints:
  - No monetary transfers.
  - Fairness

$\Rightarrow$ Random assignment

Overview

- Social planner wants to assign indivisible objects to agents.
- Examples:
  - School choice problems (NYC, Boston).
  - House allocation in colleges.
  - Course allocation.
- Typical constraints:
  - No monetary transfers.
  - Fairness

$\Rightarrow$ Random assignment

Overview

- Social planner wants to assign indivisible objects to agents.
- Examples:
  - School choice problems (NYC, Boston).
  - House allocation in colleges.
  - Course allocation.
- Typical constraints:
  - No monetary transfers.
  - Fairness

$\Rightarrow$ Random assignment

Overview

- Social planner wants to assign indivisible objects to agents.
- Examples:
  - School choice problems (NYC, Boston).
  - House allocation in colleges.
  - Course allocation.
- Typical constraints:
  - No monetary transfers.
  - Fairness

$\Rightarrow$ Random assignment

Overview

- Social planner wants to assign indivisible objects to agents.
- Examples:
  - School choice problems (NYC, Boston).
  - House allocation in colleges.
  - Course allocation.
- Typical constraints:
  - No monetary transfers.
  - Fairness

$\Rightarrow$ Random assignment

Overview

- Social planner wants to assign indivisible objects to agents.
- Examples:
  - School choice problems (NYC, Boston).
  - House allocation in colleges.
  - Course allocation.
- Typical constraints:
  - No monetary transfers.
  - Fairness

$\Rightarrow$ Random assignment

Random Assignment

- Assign four goods a, b, c, d to agents $1, 2, 3, 4$,
  1 and 2 like a, b, c, d (in this order),
  3 and 4 like b, a, c, d.
- Consider the **random priority** mechanism (agents receive goods following a randomly determined order):

	Good a	**Good** b	**Good** c	**Good** d
Agents 1,2	$5/12 \rightarrow 1/2$	$1/12 \rightarrow 0$	$1/4$	$1/4$
Agents 3,4	$1/12 \rightarrow 0$	$5/12 \rightarrow 1/2$	$1/4$	$1/4$

  The random assignment in red is preferred by everyone.
- There is even a mechanism (called PS) to find the latter assignment in this example (and improve efficiency more generally), and possibly other mechanisms.
- Can we "implement" the latter random assignment? Can we find lotteries over deterministic outcomes inducing the random assignment?

Random Assignment

- Assign four goods a, b, c, d to agents $1, 2, 3, 4$,
  1 and 2 like a, b, c, d (in this order),
  3 and 4 like b, a, c, d.
- Consider the **random priority** mechanism (agents receive goods following a randomly determined order):

	Good a	Good b	Good c	Good d
Agents 1,2	$5/12 \to 1/2$	$1/12 \to 0$	$1/4$	$1/4$
Agents 3,4	$1/12 \to 0$	$5/12 \to 1/2$	$1/4$	$1/4$

  The random assignment in red is preferred by everyone.
- There is even a mechanism (called PS) to find the latter assignment in this example (and improve efficiency more generally), and possibly other mechanisms.
- Can we "implement" the latter random assignment? Can we find lotteries over deterministic outcomes inducing the random assignment?

Random Assignment

- Assign four goods a, b, c, d to agents $1, 2, 3, 4$,
  1 and 2 like a, b, c, d (in this order),
  3 and 4 like b, a, c, d.
- Consider the **random priority** mechanism (agents receive goods following a randomly determined order):

	Good a	**Good** b	**Good** c	**Good** d
Agents 1,2	5/12	1/12	1/4	1/4
Agents 3,4	1/12	5/12	1/4	1/4

The random assignment in red is preferred by everyone.

- There is even a mechanism (called PS) to find the latter assignment in this example (and improve efficiency more generally), and possibly other mechanisms.
- Can we "implement" the latter random assignment? Can we find lotteries over deterministic outcomes inducing the random assignment?

Random Assignment

- Assign four goods a, b, c, d to agents $1, 2, 3, 4$,
  1 and 2 like a, b, c, d (in this order),
  3 and 4 like b, a, c, d.
- Consider the **random priority** mechanism (agents receive goods following a randomly determined order):

	Good a	**Good** b	**Good** c	**Good** d
Agents 1,2	$5/12 \rightarrow 1/2$	$1/12 \rightarrow 0$	$1/4$	$1/4$
Agents 3,4	$1/12 \rightarrow 0$	$5/12 \rightarrow 1/2$	$1/4$	$1/4$

  The random assignment in red is preferred by everyone.
- There is even a mechanism (called PS) to find the latter assignment in this example (and improve efficiency more generally), and possibly other mechanisms.
- Can we "implement" the latter random assignment? Can we find lotteries over deterministic outcomes inducing the random assignment?

Random Assignment

- Assign four goods a, b, c, d to agents $1, 2, 3, 4$,
  1 and 2 like a, b, c, d (in this order),
  3 and 4 like b, a, c, d.
- Consider the **random priority** mechanism (agents receive goods following a randomly determined order):

	Good a	**Good** b	**Good** c	**Good** d
Agents 1,2	$5/12 \rightarrow 1/2$	$1/12 \rightarrow 0$	$1/4$	$1/4$
Agents 3,4	$1/12 \rightarrow 0$	$5/12 \rightarrow 1/2$	$1/4$	$1/4$

  The random assignment in red is preferred by everyone.
- There is even a mechanism (called PS) to find the latter assignment in this example (and improve efficiency more generally), and possibly other mechanisms.
- Can we "implement" the latter random assignment? Can we find lotteries over deterministic outcomes inducing the random assignment?

Random Assignment

- Assign four goods a, b, c, d to agents $1, 2, 3, 4$,
  1 and 2 like a, b, c, d (in this order),
  3 and 4 like b, a, c, d.
- Consider the **random priority** mechanism (agents receive goods following a randomly determined order):

	Good a	**Good** b	**Good** c	**Good** d
Agents 1,2	$5/12 \rightarrow 1/2$	$1/12 \rightarrow 0$	$1/4$	$1/4$
Agents 3,4	$1/12 \rightarrow 0$	$5/12 \rightarrow 1/2$	$1/4$	$1/4$

  The random assignment in red is preferred by everyone.
- There is even a mechanism (called PS) to find the latter assignment in this example (and improve efficiency more generally), and possibly other mechanisms.
- Can we "implement" the latter random assignment? Can we find lotteries over deterministic outcomes inducing the random assignment?

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to **P**? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose **P** directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- It is useful to describe a random assignment by a matrix $\mathbf{P} = (P_{ia})$ where P_{ia} is the probability that i gets good a. When each of n agents receive exactly one of n goods each,
- P should be a **bistochastic matrix**, i.e.,
  - Each entry P_{ia} is nonnegative.
  - Each row sums up to one (an agent must receive a good).
  - Each column sums up to one (each object must be assigned).
- Each deterministic assignment corresponds to a **permutation matrix**, i.e., $\{0, 1\}$-valued bistochastic matrix.
- Can any random assignment be "implemented"? Does there exist a system of lotteries resolving the uncertainty according to $\mathbf{P}$? Mathematically, can any bistochastic matrix be written as a convex combination of permutation matrices?
- Implementation is important since some desirable mechanisms choose $\mathbf{P}$ directly (e.g., probabilistic serial (PS) mechanism)

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 1 & & \\ & & \\ & & \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \\ & & \\ & & \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \\ & 1 & \\ & & \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \\ & 1 & 0 \\ & & \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \\ & 1 & 0 \\ & & 1 \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Implementing Random Assignments

- Implementing random assignments is nontrivial since assignments need to be "correlated." Consider assigning 3 goods a, b, c to 3 agents $1, 2, 3$.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

- Birkhoff-von Neumann Theorem shows: Any bistochastic matrix can be written as a convex combination of permutation matrices. So, any random assignment can be implemented as a lottery over deterministic assignments when assigning n goods to n agents, with each agent getting exactly one good.

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

Can we handle complex real world constraints?

- In many applications, there are more complicated constraints.
  - Many-to-one assignment: Multiple seats in each school.
    $\Rightarrow$ Constraint may be an integer different from one
  - Non-assignment: Opt out to private school.
    $\Rightarrow$ Constraint may be an inequality, not equality
  - Group-specific quota ("Controlled choice"): Affirmative action, Gender Balance, Test score balance, District Favoritism
    $\Rightarrow$ Sub-column constraint
  - Flexible capacity: the relative sizes of alternative programs across schools or within each school may be adjustible.
    $\Rightarrow$ Multi-column constraint

What we do

- We consider a general model. We allow constraints to be placed on arbitrary subsets of entries of **P**.
- We identify a condition that is
  - sufficient for implementing random assignment, and
  - also necessary under a mild presumption.
- We apply the result to various mechanism design problems (single- and multiple goods allocation, two-sided matching, etc.)

What we do

- We consider a general model. We allow constraints to be placed on arbitrary subsets of entries of $\mathbf{P}$.
- We identify a condition that is
  - sufficient for implementing random assignment, and
  - also necessary under a mild presumption.
- We apply the result to various mechanism design problems (single- and multiple goods allocation, two-sided matching, etc.)

What we do

- We consider a general model. We allow constraints to be placed on arbitrary subsets of entries of $\mathbf{P}$.
- We identify a condition that is
    - sufficient for implementing random assignment, and
    - also necessary under a mild presumption.
- We apply the result to various mechanism design problems (single- and multiple goods allocation, two-sided matching, etc.)

What we do

- We consider a general model. We allow constraints to be placed on arbitrary subsets of entries of $\mathbf{P}$.
- We identify a condition that is
  - sufficient for implementing random assignment, and
  - also necessary under a mild presumption.
- We apply the result to various mechanism design problems (single- and multiple goods allocation, two-sided matching, etc.)

What we do

- We consider a general model. We allow constraints to be placed on arbitrary subsets of entries of $\mathbf{P}$.
- We identify a condition that is
  - sufficient for implementing random assignment, and
  - also necessary under a mild presumption.
- We apply the result to various mechanism design problems (single- and multiple goods allocation, two-sided matching, etc.)

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N| \times |O|}$.
  - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N \times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
  - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a) \in S} P_{ia} \leq \overline{q}_S,$$

    for each $S \in \mathcal{H}$.

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N|\times|O|}$.
    - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N\times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
    - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S,$$

for each $S \in \mathcal{H}$.

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N| \times |O|}$.
  - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N \times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
  - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a) \in S} P_{ia} \leq \overline{q}_S,$$

for each $S \in \mathcal{H}$.

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N| \times |O|}$.
  - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N \times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
  - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a) \in S} P_{ia} \leq \overline{q}_S,$$

for each $S \in \mathcal{H}$.

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N|\times|O|}$.
  - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N\times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
  - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S,$$

for each $S \in \mathcal{H}$.

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N| \times |O|}$.
  - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N \times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
  - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a) \in S} P_{ia} \leq \overline{q}_S,$$

for each $S \in \mathcal{H}$.

Model

- N, O are the sets of agents and goods,
- A generalized random assignment is a matrix $P = (P_{ia}) \in \mathbb{R}^{|N| \times |O|}$.
  - We allow for negative values for P_{ia} (e.g., agent i supplying good a).
- $\mathcal{H} \subset 2^{N \times O}$ is a collection of subsets of $N \times O$.
- Integers $\underline{q}_S \leq \overline{q}_S$ for each $S \in \mathcal{H}$.
  - Each set $S \in \mathcal{H}$ is understood to be a "constraint set," that is, a set of elements on which a constraint is imposed. $\underline{q}_S$ and $\overline{q}_S$ are floor and ceiling (minimum and maximum) constraints, respectively. That is, we will consider random assignment P satisfying

$$\underline{q}_S \leq \sum_{(i,a) \in S} P_{ia} \leq \overline{q}_S,$$

    for each $S \in \mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$,there exist $\alpha^1, ..., \alpha^K$ and $P^1, ..., P^K$ such that
  1. $P = \sum_{k=1}^{K} \alpha^k P^k$,
  2. $\alpha^k > 0$, $k = 1, ..., K$, and $\sum_{k=1}^{K} \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, ..., K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that"Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$, there exist $\alpha^1, ..., \alpha^K$ and $P^1, ..., P^K$ such that
  1. $P = \sum_{k=1}^K \alpha^k P^k$,
  2. $\alpha^k > 0$, $k = 1, ..., K$, and $\sum_{k=1}^K \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, ..., K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that "Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$,there exist $\alpha^1, ..., \alpha^K$ and $P^1, ..., P^K$ such that
  1. $P = \sum_{k=1}^K \alpha^k P^k$,
  2. $\alpha^k > 0$, $k = 1, ..., K$, and $\sum_{k=1}^K \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, ..., K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that "Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$,there exist $\alpha^1, ..., \alpha^K$ and $P^1, ..., P^K$ such that
  1. $P = \sum_{k=1}^{K} \alpha^k P^k$,
  2. $\alpha^k > 0$, $k = 1, ..., K$, and $\sum_{k=1}^{K} \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, ..., K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that "Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$,there exist $\alpha^1, \ldots, \alpha^K$ and $P^1, \ldots, P^K$ such that
  1. $P = \sum_{k=1}^K \alpha^k P^k$,
  2. $\alpha^k > 0$, $k = 1, \ldots, K$, and $\sum_{k=1}^K \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, \ldots, K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that "Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$,there exist $\alpha^1, ..., \alpha^K$ and $P^1, ..., P^K$ such that
  1. $P = \sum_{k=1}^K \alpha^k P^k$,
  2. $\alpha^k > 0, k = 1, ..., K$, and $\sum_{k=1}^K \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, ..., K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that"Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

BvN Decomposion

- The constraint structure $\mathcal{H}$ is **BvN decomposable** if, for each $(\underline{q}_S, \overline{q}_S)_{S\in\mathcal{H}}$ and P with $\underline{q}_S \leq \sum_{(i,a)\in S} P_{ia} \leq \overline{q}_S$ for all $S \in \mathcal{H}$,there exist $\alpha^1, ..., \alpha^K$ and $P^1, ..., P^K$ such that
  1. $P = \sum_{k=1}^K \alpha^k P^k$,
  2. $\alpha^k > 0, k = 1, ..., K$, and $\sum_{k=1}^K \alpha^k = 1$,
  3. P^k is integer-valued,
  4. for each $k = 1, ..., K$ and $S \in \mathcal{H}$,

$$\underline{q}_S \leq \sum_{(i,a)\in S} P^k_{ia} \leq \overline{q}_S.$$

- BvN decomposability means that "Every P satisfying all the given constraints in $\mathcal{H}$ can be expressed as a convex combination of integral matrices satisfying the constraints." In other words, Any random assignment satisfying constraints in $\mathcal{H}$ can be implemented as a lottery over feasible outcomes that respects constraints in $\mathcal{H}$.

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Not a Hierarchy

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Hierarchy Not a Hierarchy

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N\times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} \boxed{P_{1a} \quad P_{1b} \quad P_{1c}} \\ \\ P_{2a} \quad P_{2b} \quad P_{2c} \\ \\ P_{3a} \quad P_{3b} \quad P_{3c} \end{pmatrix}$$

Hierarchy Not a Hierarchy

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} \boxed{\boxed{P_{1a} \quad P_{1b}} \quad P_{1c}} \\ P_{2a} \quad P_{2b} \quad P_{2c} \\ P_{3a} \quad P_{3b} \quad P_{3c} \end{pmatrix}$$

Hierarchy Not a Hierarchy

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Hierarchy Not a Hierarchy

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Hierarchy Not a Hierarchy

Hierarchy

- The structure of constraint sets $\mathcal{H}$ will prove crucial for BvN decomposability.
- $\mathcal{H} \subseteq 2^{N \times O}$ is a **hierarchy** if $S \cap S' = \emptyset$ or $S \subset S'$ or $S' \subset S$ for any $S, S' \in \mathcal{H}$.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Hierarchy Not a Hierarchy

Decomposition Theorem

- $\mathcal{H} \subseteq 2^{N \times O}$ is a **bihierarchy** if it can be partitioned into two hierarchiers.

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

- Random assignments can be implemented as long as the constraint sets can be divided into two hierarchies.

Decomposition Theorem

- $\mathcal{H} \subseteq 2^{N \times O}$ is a **bihierarchy** if it can be partitioned into two hierarchiers.

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

- Random assignments can be implemented as long as the constraint sets can be divided into two hierarchies.

Decomposition Theorem

- $\mathcal{H} \subseteq 2^{N \times O}$ is a **bihierarchy** if it can be partitioned into two hierarchiers.

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

- Random assignments can be implemented as long as the constraint sets can be divided into two hierarchies.

Example: Classical One to One Assignment

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

The Birkhoff-von Neumann Theorem is a corollary of the Theorem.

Example: Classical One to One Assignment

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

The Birkhoff-von Neumann Theorem is a corollary of the Theorem.

Example: Flexible Capacities

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose a and b are two programs within a school; each program has maximum capacity of 2, and the school has maximum capacity of 3.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Example: Flexible Capacities

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose a and b are two programs within a school; each program has maximum capacity of 2, and the school has maximum capacity of 3.

$$\mathbf{P} = \begin{pmatrix} \boxed{P_{1a}} & \boxed{P_{1b}} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Example: Flexible Capacities

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose a and b are two programs within a school; each program has maximum capacity of 2, and the school has maximum capacity of 3.

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Example: Group Specific Quota

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose students 1 and 2 are ethnic majority, and 2 and 3 are male. If school a has a limit on ethnic majority while school b has a limit on male,

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Example: Group Specific Quota

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose students 1 and 2 are ethnic majority, and 2 and 3 are male. If school a has a limit on ethnic majority while school b has a limit on male,

$$\mathbf{P} = \begin{pmatrix} \boxed{\begin{matrix} P_{1a} \\ P_{2a} \\ P_{3a} \end{matrix}} & \boxed{\begin{matrix} P_{1b} \\ P_{2b} \\ P_{3b} \end{matrix}} & \boxed{\begin{matrix} P_{1c} \\ P_{2c} \\ P_{3c} \end{matrix}} \end{pmatrix}$$

Example: Group Specific Quota

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose students 1 and 2 are ethnic majority, and 2 and 3 are male. If school a has a limit on ethnic majority while school b has a limit on male,

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

Example: Group Specific Quota

Theorem

If $\mathcal{H}$ forms a bihierarchy, then it is BvN decomposable.

Suppose students 1 and 2 are ethnic majority, and 2 and 3 are male. If school a has a limit on ethnic majority while school b has a limit on male,

$$\mathbf{P} = \begin{pmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \end{pmatrix}$$

- What can go wrong without bihierarchy?
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- What can go wrong without bihierarchy?
    - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- What can go wrong without bihierarchy?
  - 2 goods and 2 agents,

    $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- What can go wrong without bihierarchy?
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- What can go wrong without bihierarchy?
  - 2 goods and 2 agents,

    $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- What can go wrong without bihierarchy?
  - 2 goods and 2 agents,

    $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- What can go wrong without bihierarchy?
  - 2 goods and 2 agents,

    $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,b),(2,a)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} =? \begin{pmatrix} 1 & 0 \\ 1 & \end{pmatrix}$$

- Bihierarchy is not necessary in general.
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,a),(2,b)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

- Bihierarchy is not necessary in general.
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,a),(2,b)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

- Bihierarchy is not necessary in general.
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,a),(2,b)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

- Bihierarchy is not necessary in general.
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,a),(2,b)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

- Bihierarchy is not necessary in general.
  - 2 goods and 2 agents, $\mathcal{H} = \{\{(1,a),(1,b)\},\{(1,a),(2,a)\},\{(1,a),(2,b)\}\}$, with each constraint being one.

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

The previous examples suggest that bihierarchies are important but not quite necessary in general.

Theorem

$\mathcal{H}$ is not BvN decomposable if $\mathcal{H}$ is not bihierarchical and $\mathcal{H}$ contains all the sets of the form

$$\{i\} \times O \text{ ("row constraints") and}$$

$$N \times \{a\} \text{ ("column constraints").}$$

The proof is constructive: we can always find a matrix P such that any decomposition attempt violates one of the constraints (as in a previous example).

In many applications in mind, row and column constraints are present. If this is the case, a bihierarchical structure is necessary for BvN decomposition.

The previous examples suggest that bihierarchies are important but not quite necessary in general.

Theorem

$\mathcal{H}$ is not BvN decomposable if $\mathcal{H}$ is not bihierarchical and $\mathcal{H}$ contains all the sets of the form

$$\{i\} \times O \text{ ("row constraints") and}$$

$$N \times \{a\} \text{ ("column constraints").}$$

The proof is constructive: we can always find a matrix P such that any decomposition attempt violates one of the constraints (as in a previous example).

In many applications in mind, row and column constraints are present. If this is the case, a bihierarchical structure is necessary for BvN decomposition.

The previous examples suggest that bihierarchies are important but not quite necessary in general.

Theorem

$\mathcal{H}$ is not BvN decomposable if $\mathcal{H}$ is not bihierarchical and $\mathcal{H}$ contains all the sets of the form

$$\{i\} \times O \text{ ("row constraints") and}$$

$$N \times \{a\} \text{ ("column constraints").}$$

The proof is constructive: we can always find a matrix P such that any decomposition attempt violates one of the constraints (as in a previous example).

In many applications in mind, row and column constraints are present. If this is the case, a bihierarchical structure is necessary for BvN decomposition.

The previous examples suggest that bihierarchies are important but not quite necessary in general.

Theorem

$\mathcal{H}$ is not BvN decomposable if $\mathcal{H}$ is not bihierarchical and $\mathcal{H}$ contains all the sets of the form

$$\{i\} \times O \text{ ("row constraints") and}$$

$$N \times \{a\} \text{ ("column constraints").}$$

The proof is constructive: we can always find a matrix P such that any decomposition attempt violates one of the constraints (as in a previous example).

In many applications in mind, row and column constraints are present. If this is the case, a bihierarchical structure is necessary for BvN decomposition.

The previous examples suggest that bihierarchies are important but not quite necessary in general.

Theorem

$\mathcal{H}$ is not BvN decomposable if $\mathcal{H}$ is not bihierarchical and $\mathcal{H}$ contains all the sets of the form

$$\{i\} \times O \text{ (“row constraints”) and}$$

$$N \times \{a\} \text{ (“column constraints”).}$$

The proof is constructive: we can always find a matrix P such that any decomposition attempt violates one of the constraints (as in a previous example).

In many applications in mind, row and column constraints are present. If this is the case, a bihierarchical structure is necessary for BvN decomposition.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a) \in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a)\in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a)\in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a)\in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a)\in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a)\in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Application: Single-Unit Assignment

- Social planner needs to assign at most one object to each agent (e.g., school choice, housing allocation).
- Each agent has strict preferences over O.
- Some additional constraints are allowed; affirmative action constraints, flexible capacity, etc.
- The situation can be modeled by a bihierarchical collection $\mathcal{H}$ such that
  - $\mathcal{H}$ contains the sets of the form $\{i\} \times O$ (row constraints).
  - the assignment $\sum_{(i,a)\in S} P_{ia}$ must not exceed some integer-valued capacity, for each $S \in \mathcal{H}$.
- **Random priority** (RP) mechanism: randomly order agents, and let each agent receive the favorite remaining good following the order, subject to the constraints described above.

Inefficiency of RP revisited

Let $N = \{1, 2, 3, 4\}$, $O = \{a, b, c, \varnothing\}$. Each good has quota of one, and only two out of three goods can actually be produced.

1 and 2 like $a, b, \varnothing$ (in this order),

3 and 4 like $c, b, \varnothing$.

RP produces random assignment:

$$RP = \begin{pmatrix} 5/12 & 1/12 & 0 & 1/2 \\ 5/12 & 1/12 & 0 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \end{pmatrix}.$$

Everyone prefers

$$P' = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix}.$$

Inefficiency of RP revisited

Let $N = \{1, 2, 3, 4\}$, $O = \{a, b, c, \emptyset\}$. Each good has quota of one, and only two out of three goods can actually be produced.

1 and 2 like $a, b, \emptyset$ (in this order),

3 and 4 like $c, b, \emptyset$.

RP produces random assignment:

$$RP = \begin{pmatrix} 5/12 & 1/12 & 0 & 1/2 \\ 5/12 & 1/12 & 0 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \end{pmatrix}.$$

Everyone prefers

$$P' = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix}.$$

Inefficiency of RP revisited

Let $N = \{1,2,3,4\}$, $O = \{a,b,c,\varnothing\}$. Each good has quota of one, and only two out of three goods can actually be produced.
1 and 2 like $a, b, \varnothing$ (in this order),
3 and 4 like $c, b, \varnothing$.
RP produces random assignment:

$$RP = \begin{pmatrix} 5/12 & 1/12 & 0 & 1/2 \\ 5/12 & 1/12 & 0 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \end{pmatrix}.$$

Everyone prefers

$$P' = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix}.$$

Inefficiency of RP revisited

Let $N = \{1,2,3,4\}$, $O = \{a,b,c,\emptyset\}$. Each good has quota of one, and only two out of three goods can actually be produced.

1 and 2 like $a, b, \emptyset$ (in this order),

3 and 4 like $c, b, \emptyset$.

RP produces random assignment:

$$RP = \begin{pmatrix} 5/12 & 1/12 & 0 & 1/2 \\ 5/12 & 1/12 & 0 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \\ 0 & 1/12 & 5/12 & 1/2 \end{pmatrix}.$$

Everyone prefers

$$P' = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix}.$$

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as “divisible” in probability units. Time runs continuously from 0 to 1, and each agent simultaneously “eats” the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as “divisible” in probability units. Time runs continuously from 0 to 1, and each agent simultaneously “eats” the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as “divisible” in probability units. Time runs continuously from 0 to 1, and each agent simultaneously “eats” the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as “divisible” in probability units. Time runs continuously from 0 to 1, and each agent simultaneously “eats” the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Probabilistic Serial Mechanism (Bogomolnaia and Moulin)

- The agents regard the goods as "divisible" in probability units. Time runs continuously from 0 to 1, and each agent simultaneously "eats" the favorite available good at speed one at each moment of time.
- The end outcome is a random assignment.
  $\Rightarrow$ Need BvN decomposition for implementation.
- Ordinally efficient in the simple classical setting.
- In the presence of extra constraints
  - Algorithm well defined? $\Rightarrow$ We define it.
  - BvN decomposition? $\Rightarrow$ Our decomposition theorem.
  - Ordinal efficiency? $\Rightarrow$ We prove this.

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

  can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

  can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

  can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

  can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

can be decomposed as

$$= \frac{1}{2} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

can also be decomposed as

$$= \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

  can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

  can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.
  A random assignment

  $$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

  can be decomposed as

  $$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

  can also be decomposed as

  $$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

- Suppose agents may be assigned to multiple objects, and they have linear preferences in the values of assigned objects, $\{v_{ia}\}$.
- There are multiple ways to implement a random assignment, some less fair than others.
- Example: $N = \{1, 2\}$; $O = \{a, b, c, d\}$, both agents like a, b, c, d; each agent demands 2 units.

  A random assignment

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

  can be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

  can also be decomposed as

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Application: Multi-Unit Assignment with Ex Post Fairness

Theorem: One-sided utility guarantee

Given any random assignment $\mathbf{P} = (P_{ia})$, there exists a BvN decomposition of $\mathbf{P}$ such that, for each $i \in N$, each ex post assignment in the decomposition gives i the expected utility within $\overline{v}_i = \max\{v_{ia} | a \in O, P_{ia} > 0\}$ of that under $\mathbf{P}$.

Proof Idea

Add a hierarchical set of "artificial" constraints in a way that bounds the extent to which each agent's utility can vary over different resolutions of the random assignment.

	a	b	c	d
1	0.5	0.5	0.5	0.5
2	0.5	0.5	0.5	0.5

This method works for more general (heterogenous preferences) cases.

Proof Idea

Add a hierarchical set of "artificial" constraints in a way that bounds the extent to which each agent's utility can vary over different resolutions of the random assignment.

	a	b	c	d
1	0.5	0.5	0.5	0.5
2	0.5	0.5	0.5	0.5

This method works for more general (heterogenous preferences) cases.

Proof Idea

Add a hierarchical set of "artificial" constraints in a way that bounds the extent to which each agent's utility can vary over different resolutions of the random assignment.

	a	b	c	d
1	0.5	0.5	0.5	0.5
2	0.5	0.5	0.5	0.5

This method works for more general (heterogenous preferences) cases.

Other Applications

- Two-Sided Matching
- Maximin Approach to Fair Division ("Santa Claus Problem")
- Scheduling Jobs on Parallel Machines: Minimize Makespan Problem
- Optimal Assignment Problem (Milgrom, 2008).
- Generalizing Hylland-Zeckhauser's pseudo market mechanism (including multi-unit demand cases, e.g., Budish 2009).

Other Applications

- Two-Sided Matching
- Maximin Approach to Fair Division ("Santa Claus Problem")
- Scheduling Jobs on Parallel Machines: Minimize Makespan Problem
- Optimal Assignment Problem (Milgrom, 2008).
- Generalizing Hylland-Zeckhauser's pseudo market mechanism (including multi-unit demand cases, e.g., Budish 2009).

Other Applications

- Two-Sided Matching
- Maximin Approach to Fair Division ("Santa Claus Problem")
- Scheduling Jobs on Parallel Machines: Minimize Makespan Problem
- Optimal Assignment Problem (Milgrom, 2008).
- Generalizing Hylland-Zeckhauser's pseudo market mechanism (including multi-unit demand cases, e.g., Budish 2009).

Other Applications

- Two-Sided Matching
- Maximin Approach to Fair Division ("Santa Claus Problem")
- Scheduling Jobs on Parallel Machines: Minimize Makespan Problem
- Optimal Assignment Problem (Milgrom, 2008).
- Generalizing Hylland-Zeckhauser's pseudo market mechanism (including multi-unit demand cases, e.g., Budish 2009).

Conclusion

Good mechanisms often find random assignments. Random assignments need to be implemented as lotteries over deterministic assignments.

We offered a mathematical result that guarantees implementation even when complicated constraints exist.

The result played a key role in applications in single- and multi-unit demand goods assignment as well as two-sided matching.

Future research:

- More applications of the result.
- Design of solutions in the absence of bihierarchical structures.

Conclusion

Good mechanisms often find random assignments. Random assignments need to be implemented as lotteries over deterministic assignments.

We offered a mathematical result that guarantees implementation even when complicated constraints exist.

The result played a key role in applications in single- and multi-unit demand goods assignment as well as two-sided matching.

Future research:

- More applications of the result.
- Design of solutions in the absence of bihierarchical structures.

Conclusion

Good mechanisms often find random assignments. Random assignments need to be implemented as lotteries over deterministic assignments.

We offered a mathematical result that guarantees implementation even when complicated constraints exist.

The result played a key role in applications in single- and multi-unit demand goods assignment as well as two-sided matching.

Future research:

- More applications of the result.
- Design of solutions in the absence of bihierarchical structures.

Conclusion

Good mechanisms often find random assignments. Random assignments need to be implemented as lotteries over deterministic assignments.

We offered a mathematical result that guarantees implementation even when complicated constraints exist.

The result played a key role in applications in single- and multi-unit demand goods assignment as well as two-sided matching.

Future research:

- More applications of the result.
- Design of solutions in the absence of bihierarchical structures.

Conclusion

Good mechanisms often find random assignments. Random assignments need to be implemented as lotteries over deterministic assignments.

We offered a mathematical result that guarantees implementation even when complicated constraints exist.

The result played a key role in applications in single- and multi-unit demand goods assignment as well as two-sided matching.

Future research:

- More applications of the result.
- Design of solutions in the absence of bihierarchical structures.

Conclusion

Good mechanisms often find random assignments. Random assignments need to be implemented as lotteries over deterministic assignments.

We offered a mathematical result that guarantees implementation even when complicated constraints exist.

The result played a key role in applications in single- and multi-unit demand goods assignment as well as two-sided matching.

Future research:

- More applications of the result.
- Design of solutions in the absence of bihierarchical structures.