Lecture 3: Assignment markets

Thilo Klein

ZEW Mannheim

Introduction

Two-sided matching models consider situations where individuals (or agents) have to be matched to other individuals (or agents).

The insights and tools of two-sided matching models can also be used to study assignment problems:

- There is a set of individuals: $I = \{i_1, i_2, \ldots, i_n\}$.
- There is a set of objects: $K = \{k_1, k_2, \ldots, k_m\}$.

The crucial difference with matching models is that while individuals have preferences over objects, these latter do not have preferences.

Introduction

Two-sided matching models consider situations where individuals (or agents) have to be matched to other individuals (or agents).

The insights and tools of two-sided matching models can also be used to study assignment problems:

- There is a set of individuals: $I = \{i_1, i_2, \ldots, i_n\}$.
- There is a set of objects: $K = \{k_1, k_2, \ldots, k_m\}$.

The crucial difference with matching models is that while individuals have preferences over objects, these latter do not have preferences.

Introduction

Two-sided matching models consider situations where individuals (or agents) have to be matched to other individuals (or agents).

The insights and tools of two-sided matching models can also be used to study assignment problems:

- There is a set of individuals: $I = \{i_1, i_2, \ldots, i_n\}$.
- There is a set of objects: $K = \{k_1, k_2, \ldots, k_m\}$.

The crucial difference with matching models is that while individuals have preferences over objects, these latter do not have preferences.

Assignments

A one-to-one assignment is defined like a matching: it specifies, who is assigned to what.

Definition

An assignment is a function $\mu : I \cup K \to I \cup K$ such that:

- for each individual $i \in I$, $\mu(i) \in K \cup \{\varnothing\}$;
- for each object $k \in K$, $\mu(k) \in I \cup \{\varnothing\}$;
- $\mu(i) = k$ if, and only if $\mu(k) = i$.

Endowments

There are two broad families of assignment problems, depending on who owns, at the outset, the objects.

- **None of the objects belong to anyone**

  This case is called the public endowment problem. All the objects belong to the whole society.

  Example: Assignment of public housing.

- **The individuals own the objects**

  This case is called the private endowment problem.

  Example: barter, with individuals trading goods without monetary transactions.

Endowments

There are two broad families of assignment problems, depending on who owns, at the outset, the objects.

- **None of the objects belong to anyone**

  This case is called the public endowment problem. All the objects belong to the whole society.

  Example: Assignment of public housing.

- **The individuals own the objects**

  This case is called the private endowment problem.

  Example: barter, with individuals trading goods without monetary transactions.

Endowments

There are two broad families of assignment problems, depending on who owns, at the outset, the objects.

- **None of the objects belong to anyone**

  This case is called the public endowment problem. All the objects belong to the whole society.

  Example: Assignment of public housing.

- **The individuals own the objects**

  This case is called the private endowment problem.

  Example: barter, with individuals trading goods without monetary transactions.

Some problem study a mix private-public endowments: some objects, but not all, are initially owned by some individuals.

Example: Dorms on (some) campus:

- public endowment: rooms left by recent graduates.
- private endowment: rooms occupied by sophomores, juniors and seniors.

Evaluating assignments

Defining an objective or a property about assignments can depend on how objects are defined:

- Each object has a priority ordering over agents.

  Such orderings specify which individual should be "considered" first when allocating the objects.

  Priority orderings work a *little bit* like preferences. But they are not preferences ($\Rightarrow$ they do not enter welfare analysis).

- The objects are "free": no specification about who has a higher priority or right over the objects.

Evaluating assignments

Defining an objective or a property about assignments can depend on how objects are defined:

- Each object has a priority ordering over agents.

  Such orderings specify which individual should be "considered" first when allocating the objects.

  Priority orderings work a *little bit* like preferences. But they are not preferences ($\Rightarrow$ they do not enter welfare analysis).

- The objects are "free": no specification about who has a higher priority or right over the objects.

Evaluating assignments

Defining an objective or a property about assignments can depend on how objects are defined:

- Each object has a priority ordering over agents.

  Such orderings specify which individual should be "considered" first when allocating the objects.

  Priority orderings work a *little bit* like preferences. But they are not preferences ($\Rightarrow$ they do not enter welfare analysis).

- The objects are "free": no specification about who has a higher priority or right over the objects.

Evaluating assignments

Defining an objective or a property about assignments can depend on how objects are defined:

- Each object has a priority ordering over agents.

  Such orderings specify which individual should be "considered" first when allocating the objects.

  Priority orderings work a *little bit* like preferences. But they are not preferences ($\Rightarrow$ they do not enter welfare analysis).

- The objects are "free": no specification about who has a higher priority or right over the objects.

Evaluating assignments

Defining an objective or a property about assignments can depend on how objects are defined:

- Each object has a priority ordering over agents.

  Such orderings specify which individual should be "considered" first when allocating the objects.

  Priority orderings work a *little bit* like preferences. But they are not preferences ($\Rightarrow$ they do not enter welfare analysis).

- The objects are "free": no specification about who has a higher priority or right over the objects.

Efficiency

One of the main properties considered when analyzing assignment is efficiency.

Definition

An assignment μ is efficient if there is no other assignment μ' such that:

- each individual either prefer μ' to μ or is indifferent between the two assignments (they obtain the same object).
- There is at least one individual who strictly prefers μ' to μ.

Efficiency

One of the main properties considered when analyzing assignment is efficiency.

Definition

An assignment μ is efficient if there is no other assignment μ' such that:

- each individual either prefer μ' to μ or is indifferent between the two assignments (they obtain the same object).
- There is at least one individual who strictly prefers μ' to μ.

Efficiency

One of the main properties considered when analyzing assignment is efficiency.

Definition

An assignment μ is efficient if there is no other assignment μ' such that:

- each individual either prefer μ' to μ or is indifferent between the two assignments (they obtain the same object).
- There is at least one individual who strictly prefers μ' to μ.

Efficiency

One of the main properties considered when analyzing assignment is efficiency.

Definition

An assignment μ is efficient if there is no other assignment μ' such that:

- each individual either prefer μ' to μ or is indifferent between the two assignments (they obtain the same object).
- There is at least one individual who strictly prefers μ' to μ.

Efficiency

One of the main properties considered when analyzing assignment is efficiency.

Definition

An assignment μ is efficient if there is no other assignment μ' such that:

- each individual either prefer μ' to μ or is indifferent between the two assignments (they obtain the same object).
- There is at least one individual who strictly prefers μ' to μ.

Example

- Three individuals: Alice, Bob and Carol.
- Three objects: A, B and C.

P_{Alice}	P_{Bob}	P_{Carol}
A	C	B
C	A	C
B	B	A

The assignment

$$\mu(\text{Alice}) = C,\ \mu(\text{Bob}) = A \text{ and } \mu(\text{Carol}) = B$$

is not efficient, because the (efficient) assignment

$$\mu(\text{Alice}) = A,\ \mu(\text{Bob}) = C \text{ and } \mu(\text{Carol}) = B$$

is better for both Alice and Bob (while making Carol indifferent).

Example

- Three individuals: Alice, Bob and Carol.
- Three objects: A, B and C.

P_{Alice}	P_{Bob}	P_{Carol}
A	C	B
C	A	C
B	B	A

The assignment

$$\mu(\text{Alice}) = C,\ \mu(\text{Bob}) = A \text{ and } \mu(\text{Carol}) = B$$

is not efficient, because the (efficient) assignment

$$\mu(\text{Alice}) = A,\ \mu(\text{Bob}) = C \text{ and } \mu(\text{Carol}) = B$$

is better for both Alice and Bob (while making Carol indifferent).

Finding efficient assignments

The simplest (and often used) solution is the serial dictatorship.

Step 0:

pick an order of the individuals (not necessarily random).

Step 1:

The first individual in the order is assigned her most preferred object.

Step k, $k \geq 2$:

The individual ranked k-th in the order is assigned her most preferred object among all objects except the ones taken by the first $k-1$ individuals in the order.

End: The algorithm stops when all individuals have chosen an object or when there is no object left.

Finding efficient assignments

The simplest (and often used) solution is the serial dictatorship.

Step 0:

pick an order of the individuals (not necessarily random).

Step 1:

The first individual in the order is assigned her most preferred object.

Step k, $k \geq 2$:

The individual ranked k-th in the order is assigned her most preferred object among all objects except the ones taken by the first $k-1$ individuals in the order.

End: The algorithm stops when all individuals have chosen an object or when there is no object left.

Finding efficient assignments

The simplest (and often used) solution is the serial dictatorship.

Step 0:

pick an order of the individuals (not necessarily random).

Step 1:

The first individual in the order is assigned her most preferred object.

Step k, $k \geq 2$:

The individual ranked k-th in the order is assigned her most preferred object among all objects except the ones taken by the first $k-1$ individuals in the order.

End: The algorithm stops when all individuals have chosen an object or when there is no object left.

Finding efficient assignments

The simplest (and often used) solution is the serial dictatorship.

Step 0:

pick an order of the individuals (not necessarily random).

Step 1:

The first individual in the order is assigned her most preferred object.

Step k, $k \geq 2$:

The individual ranked k-th in the order is assigned her most preferred object among all objects except the ones taken by the first $k-1$ individuals in the order.

End: The algorithm stops when all individuals have chosen an object or when there is no object left.

Finding efficient assignments

The simplest (and often used) solution is the serial dictatorship.

Step 0:

pick an order of the individuals (not necessarily random).

Step 1:

The first individual in the order is assigned her most preferred object.

Step k, $k \geq 2$:

The individual ranked k-th in the order is assigned her most preferred object among all objects except the ones taken by the first $k-1$ individuals in the order.

End: The algorithm stops when all individuals have chosen an object or when there is no object left.

Finding efficient assignments

The simplest (and often used) solution is the serial dictatorship.

Step 0:

pick an order of the individuals (not necessarily random).

Step 1:

The first individual in the order is assigned her most preferred object.

Step k, $k \geq 2$:

The individual ranked k-th in the order is assigned her most preferred object among all objects except the ones taken by the first $k-1$ individuals in the order.

End: The algorithm stops when all individuals have chosen an object or when there is no object left.

Proposition

For any order of the individuals:

- *the assignment obtained with the serial dictatorship is efficient.*
- *a preference revelation mechanism using serial dictatorship is strategyproof.*

Intuition:

- Take the $k-th$ individual. She took the most preferred object among the remaining ones.

  The only way to make her better off is assigning her an object taken by the 1st, 2nd, . . . , or the $(k-1)$-th individual.

- When it's your turn you obtain your best possible object. Nothing to lose by being truthful.

Proposition

For any order of the individuals:

- *the assignment obtained with the serial dictatorship is efficient.*
- *a preference revelation mechanism using serial dictatorship is strategyproof.*

Intuition:

- Take the $k-th$ individual. She took the most preferred object among the remaining ones.

  The only way to make her better off is assigning her an object taken by the 1st, 2nd, ..., or the $(k-1)$-th individual.

- When it's your turn you obtain your best possible object. Nothing to lose by being truthful.

Proposition

For any order of the individuals:

- *the assignment obtained with the serial dictatorship is efficient.*
- *a preference revelation mechanism using serial dictatorship is strategyproof.*

Intuition:

- Take the $k-th$ individual. She took the most preferred object among the remaining ones.

  The only way to make her better off is assigning her an object taken by the 1st, 2nd, ..., or the $(k-1)$-th individual.

- When it's your turn you obtain your best possible object. Nothing to lose by being truthful.

Proposition

For any order of the individuals:

- *the assignment obtained with the serial dictatorship is efficient.*
- *a preference revelation mechanism using serial dictatorship is strategyproof.*

Intuition:

- Take the $k-th$ individual. She took the most preferred object among the remaining ones.

  The only way to make her better off is assigning her an object taken by the 1st, 2nd, . . . , or the $(k-1)$-th individual.

- When it's your turn you obtain your best possible object. Nothing to lose by being truthful.

Proposition

For any order of the individuals:

- *the assignment obtained with the serial dictatorship is efficient.*
- *a preference revelation mechanism using serial dictatorship is strategyproof.*

Intuition:

- Take the $k-th$ individual. She took the most preferred object among the remaining ones.

  The only way to make her better off is assigning her an object taken by the 1st, 2nd, . . . , or the $(k-1)$-th individual.

- When it's your turn you obtain your best possible object. Nothing to lose by being truthful.

Trading with endowments

With private endowments we can use serial dictatorship but it creates a problem:

Someone may end up with an object less preferred than her endowment.

A way out is to allow individuals to trade their endowments. The most celebrated solution for that is the Top Trading Cycle algorithm (TTC).

The general principle of TTC is to draw a graph where individuals "points" to the object they want.

Trading with endowments

With private endowments we can use serial dictatorship but it creates a problem:

Someone may end up with an object less preferred than her endowment.

A way out is to allow individuals to trade their endowments. The most celebrated solution for that is the Top Trading Cycle algorithm (TTC).

The general principle of TTC is to draw a graph where individuals "points" to the object they want.

Trading with endowments

With private endowments we can use serial dictatorship but it creates a problem:

Someone may end up with an object less preferred than her endowment.

A way out is to allow individuals to trade their endowments. The most celebrated solution for that is the Top Trading Cycle algorithm (TTC).

The general principle of TTC is to draw a graph where individuals "points" to the object they want.

Trading with endowments

With private endowments we can use serial dictatorship but it creates a problem:

Someone may end up with an object less preferred than her endowment.

A way out is to allow individuals to trade their endowments. The most celebrated solution for that is the Top Trading Cycle algorithm (TTC).

The general principle of TTC is to draw a graph where individuals “points” to the object they want.

Trading with endowments

With private endowments we can use serial dictatorship but it creates a problem:

Someone may end up with an object less preferred than her endowment.

A way out is to allow individuals to trade their endowments. The most celebrated solution for that is the Top Trading Cycle algorithm (TTC).

The general principle of TTC is to draw a graph where individuals “points” to the object they want.

Top Trading Cycle algorithm

Step 1

Each individual points (we draw an arrow) to the individual owning the object she prefers the most (could be herself).

There is always at least one cycle: when starting from an agent and following the arrows we eventually reach the agent again.

For each agent in a cycle assign her the object owned by the individual she is pointing to.

Remove from the problem all the agents (and their objects) who were part of a cycle.

Top Trading Cycle algorithm

Step 1

Each individual points (we draw an arrow) to the individual owning the object she prefers the most (could be herself).

There is always at least one cycle: when starting from an agent and following the arrows we eventually reach the agent again.

For each agent in a cycle assign her the object owned by the individual she is pointing to.

Remove from the problem all the agents (and their objects) who were part of a cycle.

Top Trading Cycle algorithm

Step 1

Each individual points (we draw an arrow) to the individual owning the object she prefers the most (could be herself).

There is always at least one cycle: when starting from an agent and following the arrows we eventually reach the agent again.

For each agent in a cycle assign her the object owned by the individual she is pointing to.

Remove from the problem all the agents (and their objects) who were part of a cycle.

Top Trading Cycle algorithm

Step 1

Each individual points (we draw an arrow) to the individual owning the object she prefers the most (could be herself).

There is always at least one cycle: when starting from an agent and following the arrows we eventually reach the agent again.

For each agent in a cycle assign her the object owned by the individual she is pointing to.

Remove from the problem all the agents (and their objects) who were part of a cycle.

Top Trading Cycle algorithm

Step 1

Each individual points (we draw an arrow) to the individual owning the object she prefers the most (could be herself).

There is always at least one cycle: when starting from an agent and following the arrows we eventually reach the agent again.

For each agent in a cycle assign her the object owned by the individual she is pointing to.

Remove from the problem all the agents (and their objects) who were part of a cycle.

Step $\mathbf{k}, \mathbf{k} \geq \mathbf{2}$

Do like in Step 1, with individuals pointing to their most preferred object among the objects that have not been removed at an earlier step.

End:

The algorithm stops when all individuals have been removed or there are no acceptable objects left for any individual that has not been removed yet.

Step $\mathbf{k}, \mathbf{k} \geq \mathbf{2}$

Do like in Step 1, with individuals pointing to their most preferred object among the objects that have not been removed at an earlier step.

End:

The algorithm stops when all individuals have been removed or there are no acceptable objects left for any individual that has not been removed yet.

Step $\mathbf{k}, \mathbf{k} \geq \mathbf{2}$

Do like in Step 1, with individuals pointing to their most preferred object among the objects that have not been removed at an earlier step.

End:

The algorithm stops when all individuals have been removed or there are no acceptable objects left for any individual that has not been removed yet.

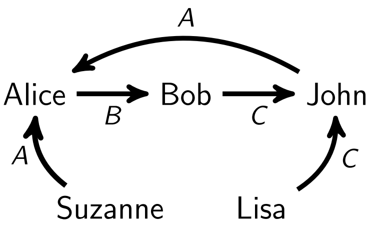
Example

Endowment →	A	B	C	D	E
	P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
	B	C	A	C	A
	E	A	E	A	C
	D	D	D	E	B
	C	E	B	B	D
	A	B	C	D	E

Example

Endowment $\rightarrow$	A	B	C	D	E
	P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
	B	C	A	C	A
	E	A	E	A	C
	D	D	D	E	B
	C	E	B	B	D
	A	B	C	D	E

Step 1



We have a cycle:

John gets A

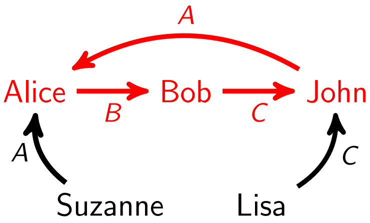
Alice gets B

Bob gets C

Example

Endowment $\rightarrow$	A	B	C	D	E
	P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
	B	C	A	C	A
	E	A	E	A	C
	D	D	D	E	B
	C	E	B	B	D
	A	B	C	D	E

Step 1



We have a cycle:

John gets A

Alice gets B

Bob gets C

Example

Endowment $\rightarrow$	A	B	C	D	E
	P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
	B	C	A	C	A
	E	A	E	A	C
	D	D	D	E	B
	C	E	B	B	D
	A	B	C	D	E

Step 2

$$\begin{tikzcd} \text{Lisa} \arrow[rr, bend left, "E"] & & \text{Suzanne} \arrow[ll, bend left, "D"] \end{tikzcd}$$

We have a cycle:

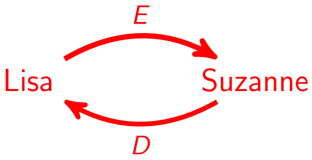
Lisa gets E

Suzanne gets D

Example

Endowment $\rightarrow$	A	B	C	D	E
	P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
	B	C	A	C	A
	E	A	E	A	C
	D	D	D	E	B
	C	E	B	B	D
	A	B	C	D	E

Step 2



We have a cycle:

Lisa gets E

Suzanne gets D

Example

Endowment $\rightarrow$	A	B	C	D	E
	P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
	B	C	A	C	A
	E	A	E	A	C
	D	D	D	E	B
	C	E	B	B	D
	A	B	C	D	E

Final allocation:

$$\mu(\text{Alice}) = B \qquad \mu(\text{Bob}) = C \qquad \mu(\text{John}) = A$$
$$\mu(\text{Lisa}) = E \qquad \mu(\text{Suzanne}) = D$$

Proposition

For any problem:

- *the assignment obtained with the Top Trading Cycle algorithm is efficient.*
- *a preference revelation mechanism using the Top Trading Cycle algorithm is strategyproof.*

The intuition for the efficiency of TTC is similar to the argument used for serial dictatorship.

- Individuals assigned in the first cycles obtain their top choices: for them, it's efficient.
- For the individuals in the second cycle, either:
  - they have their top choice;
  - their top choice is gone: assigned to someone in the first cycle. The only way to make them better off is to make a first cycle individual worse off.

Proposition

For any problem:

- *the assignment obtained with the Top Trading Cycle algorithm is efficient.*
- *a preference revelation mechanism using the Top Trading Cycle algorithm is strategyproof.*

The intuition for the efficiency of TTC is similar to the argument used for serial dictatorship.

- Individuals assigned in the first cycles obtain their top choices: for them, it's efficient.
- For the individuals in the second cycle, either:
  - they have their top choice;
  - their top choice is gone: assigned to someone in the first cycle. The only way to make them better off is to make a first cycle individual worse off.

Proposition

For any problem:

- *the assignment obtained with the Top Trading Cycle algorithm is efficient.*
- *a preference revelation mechanism using the Top Trading Cycle algorithm is strategyproof.*

The intuition for the efficiency of TTC is similar to the argument used for serial dictatorship.

- Individuals assigned in the first cycles obtain their top choices: for them, it's efficient.
- For the individuals in the second cycle, either:
  - they have their top choice;
  - their top choice is gone: assigned to someone in the first cycle. The only way to make them better off is to make a first cycle individual worse off.

Proposition

For any problem:

- *the assignment obtained with the Top Trading Cycle algorithm is efficient.*
- *a preference revelation mechanism using the Top Trading Cycle algorithm is strategyproof.*

The intuition for the efficiency of TTC is similar to the argument used for serial dictatorship.

- Individuals assigned in the first cycles obtain their top choices: for them, it's efficient.
- For the individuals in the second cycle, either:
  - they have their top choice;
  - their top choice is gone: assigned to someone in the first cycle. The only way to make them better off is to make a first cycle individual worse off.

In fact, both the serial dictatorship and TTC have stronger incentive properties: they are both group strategyproof.

But TTC is, in some sense, the "right" algorithm.

Recall that an assignment is individually rational if nobody gets an object less preferred than the endowment.

Theorem

An assignment mechanism is strategyproof, efficient and individually rational if, and only if it uses the Top Trading Cycle algorithm.

In fact, both the serial dictatorship and TTC have stronger incentive properties: they are both group strategyproof.

But TTC is, in some sense, the "right" algorithm.

Recall that an assignment is individually rational if nobody gets an object less preferred than the endowment.

Theorem

An assignment mechanism is strategyproof, efficient and individually rational if, and only if it uses the Top Trading Cycle algorithm.

In fact, both the serial dictatorship and TTC have stronger incentive properties: they are both group strategyproof.

But TTC is, in some sense, the “right” algorithm.

Recall that an assignment is individually rational if nobody gets an object less preferred than the endowment.

Theorem

An assignment mechanism is strategyproof, efficient and individually rational if, and only if it uses the Top Trading Cycle algorithm.

When objects have priorities

TTC can also be used when endowments are public and objects have priority orderings over individuals.

The algorithm needs a few tweaks. At any step,

- Each individual points to the object she wants;
- Each object points to the individual with the highest priority (among the individuals who are not yet assigned any object).

When objects have priorities

TTC can also be used when endowments are public and objects have priority orderings over individuals.

The algorithm needs a few tweaks. At any step,

- Each individual points to the object she wants;
- Each object points to the individual with the highest priority (among the individuals who are not yet assigned any object).

When objects have priorities

TTC can also be used when endowments are public and objects have priority orderings over individuals.

The algorithm needs a few tweaks. At any step,

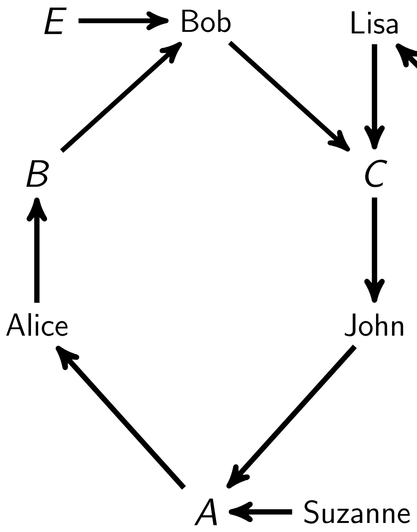
- Each individual points to the object she wants;
- Each object points to the individual with the highest priority (among the individuals who are not yet assigned any object).

Example

P_{Alice}	P_{Bob}	P_{John}	P_{Lisa}	P_{Suzanne}
B	C	A	C	A
E	A	E	A	C
D	D	D	E	B
C	E	B	B	D
A	B	C	D	E

P_A	P_B	P_C	P_D	P_E
Alice	Bob	John	Lisa	Bob
John	Lisa	Suzanne	Suzanne	Lisa
Bob	Suzanne	John	Alice	Alice
Suzanne	Alice	Lisa	John	John
Lisa	John	Alice	Bob	Suzanne

Step 1



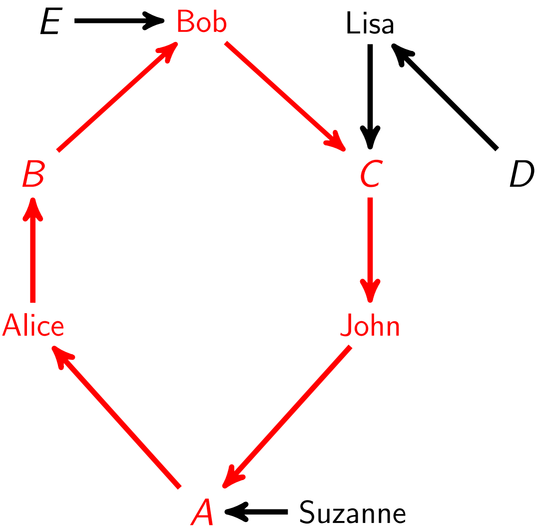
We have a cycle:

Bob gets C

John gets A

Alice gets B

Step 1



We have a cycle:

Bob gets C

John gets A

Alice gets B

Step 2

$$\begin{tikzcd} \text{Lisa} \arrow[r] & C \arrow[d] \\ D \arrow[u] & \text{Suzanne} \arrow[l] \end{tikzcd}$$

We have a cycle:

Suzanne gets D

Lisa gets C

Step 2

$$\begin{tikzcd} \text{Lisa} \arrow[r] & C \arrow[d] \\ D \arrow[u] & \text{Suzanne} \arrow[l] \end{tikzcd}$$

We have a cycle:

Suzanne gets D

Lisa gets C

Individual rationality and the core

Individual rationality is the requirement that nobody gets an assignment less preferred than the endowment.

We can extend this notion to groups of individuals.

Definition

The core of an assignment problem is the set of all assignments μ such that there is no coalition S of individuals and an assignment μ' for which:

1. For each individual in $i \in S$, the object $\mu'(i)$ is the endowment of another individual in S;
2. Each individual in S prefers μ' to μ or is indifferent between μ and μ' and there is at least one individual who strictly prefers μ' to μ.

Individual rationality and the core

Individual rationality is the requirement that nobody gets an assignment less preferred than the endowment.

We can extend this notion to groups of individuals.

Definition

The core of an assignment problem is the set of all assignments μ such that there is no coalition S of individuals and an assignment μ' for which:

1. For each individual in $i \in S$, the object $\mu'(i)$ is the endowment of another individual in S;
2. Each individual in S prefers μ' to μ or is indifferent between μ and μ' and there is at least one individual who strictly prefers μ' to μ.

Individual rationality and the core

Individual rationality is the requirement that nobody gets an assignment less preferred than the endowment.

We can extend this notion to groups of individuals.

Definition

The core of an assignment problem is the set of all assignments μ such that there is no coalition S of individuals and an assignment μ' for which:

1. For each individual in $i \in S$, the object $\mu'(i)$ is the endowment of another individual in S;
2. Each individual in S prefers μ' to μ or is indifferent between μ and μ' and there is at least one individual who strictly prefers μ' to μ.

Individual rationality and the core

Individual rationality is the requirement that nobody gets an assignment less preferred than the endowment.

We can extend this notion to groups of individuals.

Definition

The core of an assignment problem is the set of all assignments μ such that there is no coalition S of individuals and an assignment μ' for which:

1. For each individual in $i \in S$, the object $\mu'(i)$ is the endowment of another individual in S;
2. Each individual in S prefers μ' to μ or is indifferent between μ and μ' and there is at least one individual who strictly prefers μ' to μ.

Individual rationality and the core

Individual rationality is the requirement that nobody gets an assignment less preferred than the endowment.

We can extend this notion to groups of individuals.

Definition

The core of an assignment problem is the set of all assignments μ such that there is no coalition S of individuals and an assignment μ' for which:

1. For each individual in $i \in S$, the object $\mu'(i)$ is the endowment of another individual in S;
2. Each individual in S prefers μ' to μ or is indifferent between μ and μ' and there is at least one individual who strictly prefers μ' to μ.

Individual rationality and the core

Individual rationality is the requirement that nobody gets an assignment less preferred than the endowment.

We can extend this notion to groups of individuals.

Definition

The core of an assignment problem is the set of all assignments μ such that there is no coalition S of individuals and an assignment μ' for which:

1. For each individual in $i \in S$, the object $\mu'(i)$ is the endowment of another individual in S;
2. Each individual in S prefers μ' to μ or is indifferent between μ and μ' and there is at least one individual who strictly prefers μ' to μ.

Efficiency $\neq$ core

Endowment $\rightarrow$	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
	B	A	A
	C	B	B
	A	C	C

The assignment

$$\mu(\text{Alice}) = C, \qquad \mu(\text{Bob}) = B, \qquad \mu(\text{Carol}) = A$$

is efficient.But it's not in the core:

But Alice & Bob can be better off without Carol. They can trade their their endowments (Alice gets B and Bob gets A), a better option than μ.

Efficiency $\neq$ core

Endowment $\rightarrow$	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
	B	A	A
	C	B	B
	A	C	C

The assignment

$$\mu(\text{Alice}) = C, \qquad \mu(\text{Bob}) = B, \qquad \mu(\text{Carol}) = A$$

is efficient.But it's not in the core:

But Alice & Bob can be better off without Carol. They can trade their their endowments (Alice gets B and Bob gets A), a better option than μ.

Efficiency $\neq$ core

$$\begin{array}{r|ccc} \text{Endowment} \rightarrow & A & B & C \\ \hline & P_{\text{Alice}} & P_{\text{Bob}} & P_{\text{Carol}} \\ \hline & B & A & A \\ & C & B & B \\ & A & C & C \end{array}$$

The assignment

$$\mu(\text{Alice}) = C, \qquad \mu(\text{Bob}) = B, \qquad \mu(\text{Carol}) = A$$

is efficient. But it's not in the core:

But Alice & Bob can be better off without Carol. They can trade their their endowments (Alice gets B and Bob gets A), a better option than μ.

Efficiency $\neq$ core

$$
\begin{array}{r|ccc}
\text{Endowment} \rightarrow & A & B & C \\
\hline
& P_{\text{Alice}} & P_{\text{Bob}} & P_{\text{Carol}} \\
\hline
& B & A & A \\
& C & B & B \\
& A & C & C
\end{array}
$$

The assignment

$$\mu(\text{Alice}) = C, \qquad \mu(\text{Bob}) = B, \qquad \mu(\text{Carol}) = A$$

is efficient.But it's not in the core:

But Alice & Bob can be better off without Carol. They can trade their their endowments (Alice gets B and Bob gets A), a better option than μ.

Once again, TTC proves to be a really good algorithm:

Theorem (Roth and Postlewaite)

For any assignment problem:

- *there is always a unique assignment that is in the core;*
- *the core assignment can be obtained with the Top Trading Cycles algorithm.*

Mixed public-private endowments

The case of dorms on campus is a classic example of assignment with pubic and private endowments:

- **Private endowments**: the students who were already on campus the previous academic year.

  The room they occupied the previous year is their private endowment;

- **Public endowments**: the rooms that are left vacant by the students who just graduated.

  The newly arrived students do not have any endowments.

Mixed public-private endowments

The case of dorms on campus is a classic example of assignment with pubic and private endowments:

- **Private endowments**: the students who were already on campus the previous academic year.

  The room they occupied the previous year is their private endowment;

- **Public endowments**: the rooms that are left vacant by the students who just graduated.

  The newly arrived students do not have any endowments.

Mixed public-private endowments

The case of dorms on campus is a classic example of assignment with pubic and private endowments:

- **Private endowments**: the students who were already on campus the previous academic year.

  The room they occupied the previous year is their private endowment;

- **Public endowments**: the rooms that are left vacant by the students who just graduated.

  The newly arrived students do not have any endowments.

Mixed public-private endowments

The case of dorms on campus is a classic example of assignment with pubic and private endowments:

- **Private endowments**: the students who were already on campus the previous academic year.

  The room they occupied the previous year is their private endowment;

- **Public endowments**: the rooms that are left vacant by the students who just graduated.

  The newly arrived students do not have any endowments.

We review various mechanisms used on US campuses and use the following model:

- There is a set of individuals and a set of houses (the objects).
- Individuals are split into two groups:
  - Existing tenant: an individual who already "owns" a house (a student who was on campus the previous year and hasn't graduated yet).

    Their houses are private endowments.
  - New applicants: an individual who does not has any house (a freshman).

We review various mechanisms used on US campuses and use the following model:

- There is a set of individuals and a set of houses (the objects).
- Individuals are split into two groups:
  - Existing tenant: an individual who already "owns" a house (a student who was on campus the previous year and hasn't graduated yet).

    Their houses are private endowments.
  - New applicants: an individual who does not has any house (a freshman).

We review various mechanisms used on US campuses and use the following model:

- There is a set of individuals and a set of houses (the objects).
- Individuals are split into two groups:
  - Existing tenant: an individual who already "owns" a house (a student who was on campus the previous year and hasn't graduated yet).

    Their houses are private endowments.
  - New applicants: an individual who does not has any house (a freshman).

We review various mechanisms used on US campuses and use the following model:

- There is a set of individuals and a set of houses (the objects).
- Individuals are split into two groups:
  - Existing tenant: an individual who already “owns” a house (a student who was on campus the previous year and hasn’t graduated yet).

    Their houses are private endowments.
  - New applicants: an individual who does not has any house (a freshman).

We review various mechanisms used on US campuses and use the following model:

- There is a set of individuals and a set of houses (the objects).
- Individuals are split into two groups:
  - Existing tenant: an individual who already “owns” a house (a student who was on campus the previous year and hasn’t graduated yet).

    Their houses are private endowments.
  - New applicants: an individual who does not has any house (a freshman).

Carnegie-Mellon, Duke, Harvard

The mechanism used by (among others) these universities is known as the Random serial dictatorship with squatting rights.

Step 1: Existing tenants announce whether they want to keep their house. If they do so they are assigned their house, otherwise their house is added to the pool of vacant houses.

Step 2: We draw a random ordering of all individuals consisting of the new applicants and the existing tenants who chose not to keep their house.

Step 3: Rhn the Serial Dictatorship with the order chosen in Step 2.

Carnegie-Mellon, Duke, Harvard

The mechanism used by (among others) these universities is known as the Random serial dictatorship with squatting rights.

Step 1: Existing tenants announce whether they want to keep their house. If they do so they are assigned their house, otherwise their house is added to the pool of vacant houses.

Step 2: We draw a random ordering of all individuals consisting of the new applicants and the existing tenants who chose not to keep their house.

Step 3: Rhn the Serial Dictatorship with the order chosen in Step 2.

Carnegie-Mellon, Duke, Harvard

The mechanism used by (among others) these universities is known as the Random serial dictatorship with squatting rights.

Step 1: Existing tenants announce whether they want to keep their house. If they do so they are assigned their house, otherwise their house is added to the pool of vacant houses.

Step 2: We draw a random ordering of all individuals consisting of the new applicants and the existing tenants who chose not to keep their house.

Step 3: Rhn the Serial Dictatorship with the order chosen in Step 2.

Carnegie-Mellon, Duke, Harvard

The mechanism used by (among others) these universities is known as the Random serial dictatorship with squatting rights.

Step 1: Existing tenants announce whether they want to keep their house. If they do so they are assigned their house, otherwise their house is added to the pool of vacant houses.

Step 2: We draw a random ordering of all individuals consisting of the new applicants and the existing tenants who chose not to keep their house.

Step 3: Rhn the Serial Dictatorship with the order chosen in Step 2.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

  Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

  But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

  Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

  But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

  Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

  But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Endowment: Alice owns A

P_{Alice}	P_{Bob}	P_{Carol}
B	B	A
A	C	B
C	A	C

- Dictators' order: Carol, Bob, Alice.
- If all participate,

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = A$$

  Alice is better off not participating.

- Alice opts out. So we have

$$\mu'(\text{Alice}) = A \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = B$$

  But Pareto dominated by

$$\mu''(\text{Alice}) = B \quad \mu''(\text{Bob}) = C \quad \mu''(\text{Carol}) = A$$

$\Rightarrow$ This algorithm does not guarantee individually rational assignments. Risk averse tenants may prefer to opt out, resulting in inefficient assignments.

Rochester

A first solution to avoid the risk of tenants being worse off is the Random Serial Dictatorship with Waiting Lists.

At any step, an individual can only take a house that is available. This is the case when:

- the house is part of the public endowment; or
- the house is left vacant by the existing tenant.

An obtainable house is:

- any house for new applicants;
- the endowment or a preferred house for existing tenants.

Rochester

A first solution to avoid the risk of tenants being worse off is the Random Serial Dictatorship with Waiting Lists.

At any step, an individual can only take a house that is available. This is the case when:

- the house is part of the public endowment; or
- the house is left vacant by the existing tenant.

An obtainable house is:

- any house for new applicants;
- the endowment or a preferred house for existing tenants.

Rochester

A first solution to avoid the risk of tenants being worse off is the Random Serial Dictatorship with Waiting Lists.

At any step, an individual can only take a house that is available. This is the case when:

- the house is part of the public endowment; or
- the house is left vacant by the existing tenant.

An obtainable house is:

- any house for new applicants;
- the endowment or a preferred house for existing tenants.

Rochester

A first solution to avoid the risk of tenants being worse off is the Random Serial Dictatorship with Waiting Lists.

At any step, an individual can only take a house that is available. This is the case when:

- the house is part of the public endowment; or
- the house is left vacant by the existing tenant.

An obtainable house is:

- any house for new applicants;
- the endowment or a preferred house for existing tenants.

Step 1: Draw a random order of all individuals.

Step 2: only vacant houses are available.

Run the serial dictatorship. If an existing tenant selects a different house than her endowment, the endowment is added to the set of available houses.

Step k**,** $k \geq 3$: the set of available houses is constructed at the end of Step $k-1$.
Run the serial dictatorship like in Step 2 with individuals not assigned yet.

End: The algorithm ends where there is no remaining individual or when there are no available house left for any individual.

Step 1: Draw a random order of all individuals.

Step 2: only vacant houses are available.

Run the serial dictatorship. If an existing tenant selects a different house than her endowment, the endowment is added to the set of available houses.

Step k, $k \geq 3$: the set of available houses is constructed at the end of Step $k-1$.
Run the serial dictatorship like in Step 2 with individuals not assigned yet.

End: The algorithm ends where there is no remaining individual or when there are no available house left for any individual.

Step 1: Draw a random order of all individuals.

Step 2: only vacant houses are available.

Run the serial dictatorship. If an existing tenant selects a different house than her endowment, the endowment is added to the set of available houses.

Step k**,** $k \geq 3$: the set of available houses is constructed at the end of Step $k-1$.
Run the serial dictatorship like in Step 2 with individuals not assigned yet.

End: The algorithm ends where there is no remaining individual or when there are no available house left for any individual.

Step 1: Draw a random order of all individuals.

Step 2: only vacant houses are available.

Run the serial dictatorship. If an existing tenant selects a different house than her endowment, the endowment is added to the set of available houses.

Step k, $k \geq 3$: the set of available houses is constructed at the end of Step $k-1$.
Run the serial dictatorship like in Step 2 with individuals not assigned yet.

End: The algorithm ends where there is no remaining individual or when there are no available house left for any individual.

Endowment $\rightarrow$	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
Public endowment: D	B	C	A
	C	A	D
Available houses: D	A	B	C
	D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

$\Rightarrow$ this algorithm does not guarantee efficiency

Endowment $\rightarrow$	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
Public endowment: D	B	C	A
	C	A	D
Available houses: D	A	B	C
	D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

$\Rightarrow$ this algorithm does not guarantee efficiency.

Endowment →	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
Public endowment: D	B	C	A
	C	A	D
Available houses: C	A	B	C
	D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

⇒ this algorithm does not guarantee efficiency

	Endowment $\rightarrow$	A	B	C
		P_{Alice}	P_{Bob}	P_{Carol}
Public endowment:	D	B	C	A
		C	A	D
Available houses:	A	A	B	C
		D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

$\Rightarrow$ this algorithm does not guarantee efficiency

Endowment $\rightarrow$	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
Public endowment: D	B	C	A
	C	A	D
Available houses:	A	B	C
	D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

$\Rightarrow$ this algorithm does not guarantee efficiency

Endowment →	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
Public endowment: D	B	C	A
	C	A	D
Available houses:	A	B	C
	D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

⇒ this algorithm does not guarantee efficiency

Endowment $\rightarrow$	A	B	C
	P_{Alice}	P_{Bob}	P_{Carol}
Public endowment: D	B	C	A
	C	A	D
Available houses:	A	B	C
	D	D	B

Order of dictators: Alice, Bob and Chris.

- Step 2: Carol takes D (D not available for Alice and Bob).
- Step 3: C now available, Alice is first, she takes C.
- Step 4: A now available, Bob takes it.

Final assignment:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = A \quad \mu(\text{Carol}) = D$$

But Pareto dominated by

$$\mu'(\text{Alice}) = B \quad \mu'(\text{Bob}) = C \quad \mu'(\text{Carol}) = A$$

$\Rightarrow$ this algorithm does not guarantee efficiency.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

MIT-NH4

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs: the requested house has a tenant and all the remaining houses are worse for the tenant. The conflicting individual is the one who picked the tenant's house.

Step 3: If there's a squatting conflict:

- The existing tenant is assigned her house.
- All assignments starting from the conflicting individual are cancelled.
- Resume serial dictatorship starting with the conflicting individual.

End: The algorithm stops when there is no house or individual left. At this point all tentative assignments are finalized.

Endowment $\rightarrow$	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C
- Bob chooses D
- Carol chooses E
- Denis: conflict! C, E and D are taken.
  conflicting individual: Bob.
  $\Rightarrow$ Bob and Carol's assignment are cancelled and Denis is assigned his house, D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C
- Bob chooses D
- Carol chooses E
- Denis: conflict! C, E and D are taken.
  conflicting individual: Bob.
  $\Rightarrow$ Bob and Carol's assignment are cancelled and Denis is assigned his house, D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C
- Bob chooses D
- Carol chooses E
- Denis: conflict! C, E and D are taken.
  conflicting individual: Bob.
  $\Rightarrow$ Bob and Carol's assignment are cancelled and Denis is assigned his house, D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C
- Bob chooses D
- Carol chooses E
- Denis: conflict! C, E and D are taken.
  conflicting individual: Bob.
  $\Rightarrow$ Bob and Carol's assignment are cancelled and Denis is assigned his house, D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C
- Bob chooses D
- Carol chooses E
- Denis: conflict! C, E and D are taken.
  conflicting individual: Bob.
  $\Rightarrow$ Bob and Carol's assignment are cancelled and Denis is assigned his house, D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C
- Bob chooses D
- Carol chooses E
- Denis: conflict! C, E and D are taken.
  conflicting individual: Bob.
  ⇒ Bob and Carol's assignment are cancelled and Denis is assigned his house, D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Bob chooses E, the next available house
- Carol: conflict! E and C are taken.
  conflicting individual: Alice.
  $\Rightarrow$ Alice and Bob's assignment are cancelled and Carol is assigned her house, C.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Bob chooses E, the next available house
- Carol: conflict! E and C are taken.
  conflicting individual: Alice.
  $\Rightarrow$ Alice and Bob's assignment are cancelled and Carol is assigned her house, C.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Bob chooses E, the next available house
- Carol: conflict! E and C are taken.
  conflicting individual: Alice.
  $\Rightarrow$ Alice and Bob's assignment are cancelled and Carol is assigned her house, C.

	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
Endowment $\rightarrow$	A	B	C	D	
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Bob chooses E, the next available house
- Carol: conflict! E and C are taken.
  conflicting individual: Alice.
  $\Rightarrow$ Alice and Bob's assignment are cancelled and Carol is assigned her house, C.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses E, the next available house.
- Bob chooses B.
- Erin chooses A.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = E \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = C \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses E, the next available house.
- Bob chooses B.
- Erin chooses A.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = E \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = C \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses E, the next available house.
- Bob chooses B.
- Erin chooses A.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = E \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = C \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses E, the next available house.
- Bob chooses B.
- Erin chooses A.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = E \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = C \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Endowment $\rightarrow$	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

But this assignment is Pareto dominated by:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = E \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Two efficient solutions

The MIT-NH4, Rochester or Carnegie-Mellon/Duke/Harvard mechanisms show that finding an efficient and individually rational assignment is not trivial.

There is a way out, though. Use either:

- An improved version of the MIT-NH4 algorithm; or
- the Top Trading Cycle algorithm.

These solutions turn out to have an additional property: they are both strategyproof.

Two efficient solutions

The MIT-NH4, Rochester or Carnegie-Mellon/Duke/Harvard mechanisms show that finding an efficient and individually rational assignment is not trivial.

There is a way out, though. Use either:

- An improved version of the MIT-NH4 algorithm; or
- the Top Trading Cycle algorithm.

These solutions turn out to have an additional property: they are both strategyproof.

Two efficient solutions

The MIT-NH4, Rochester or Carnegie-Mellon/Duke/Harvard mechanisms show that finding an efficient and individually rational assignment is not trivial.

There is a way out, though. Use either:

- An improved version of the MIT-NH4 algorithm; or
- the Top Trading Cycle algorithm.

These solutions turn out to have an additional property: they are both strategyproof.

Two efficient solutions

The MIT-NH4, Rochester or Carnegie-Mellon/Duke/Harvard mechanisms show that finding an efficient and individually rational assignment is not trivial.

There is a way out, though. Use either:

- An improved version of the MIT-NH4 algorithm; or
- the Top Trading Cycle algorithm.

These solutions turn out to have an additional property: they are both strategyproof.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.
  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.
- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.
  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

Improving the MIT-NH4 mechanism

The issue is how conflicts defined and solved:

- **MIT-NH4**:

  If there is a conflict: all tenant's acceptable houses taken.

  - All assignments starting from the conflicting individual are cancelled;
  - Existing tenant is assigned her house;
  - Resume serial dictatorship starting from the conflicting individual.

- **You Request My House — I Get Your Turn**:

  If there is a conflict: the tenant hasn't chosen a house yet.

  - All assignments starting from the conflicting individual are cancelled;
  - Rearrange the orders of dictators: existing tenant moved just above the conflicting individual;
  - Resume serial dictatorship starting from the existing tenant.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

You Request My House — I Get Your Turn

Step 1: Draw a random order of all individuals.

Step 2: Assign tentatively houses using serial dictatorship until a squatting conflict occurs.

Step 3: It there is a squatting conflict:

- All assignments starting from the conflicting individual are cancelled.
- In the ordering of individuals move the existing tenant just above the conflicting individual.
- Resume serial dictatorship starting with the existing tenant.

Step 4: If there is a cycle, assign the houses (like in TTC) and remove the individuals and houses in the cycle from the problem.

End: The algorithm stops when there are no house or individual left.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C, but tenant (Carol) hasn't chosen a house yet.
- Carol chooses E, then Alice chooses C.
- Bob chooses D, but tenant (Denis) hasn't chosen a house yet.

New order: Carol, Alice, Denis, Bob and Erin.

- Denis chooses D. A trivial cycle, so Denis is assigned D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C, but tenant (Carol) hasn't chosen a house yet.

  New order: Carol, Alice, Bob, Denis and Erin.
- Carol chooses E, then Alice chooses C.
- Bob chooses D, but tenant (Denis) hasn't chosen a house yet.

  New order: Carol, Alice, Denis, Bob and Erin.
- Denis chooses D. A trivial cycle, so Denis is assigned D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C, but tenant (Carol) hasn't chosen a house yet.

  New order: Carol, Alice, Bob, Denis and Erin.

- Carol chooses E, then Alice chooses C.
- Bob chooses D, but tenant (Denis) hasn't chosen a house yet.

  New order: Carol, Alice, Denis, Bob and Erin.

- Denis chooses D. A trivial cycle, so Denis is assigned D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C, but tenant (Carol) hasn't chosen a house yet.

  New order: Carol, Alice, Bob, Denis and Erin.

- Carol chooses E, then Alice chooses C.
- Bob chooses D, but tenant (Denis) hasn't chosen a house yet.

  New order: Carol, Alice, Denis, Bob and Erin.

- Denis chooses D. A trivial cycle, so Denis is assigned D.

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C, but tenant (Carol) hasn't chosen a house yet.
- Carol chooses E, then Alice chooses C.
- Bob chooses D, but tenant (Denis) hasn't chosen a house yet.

New order: Carol, Alice, Denis, Bob and Erin.

- Denis chooses D. A trivial cycle, so Denis is assigned D.

Endowment $\rightarrow$	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Alice, Bob, Carol, Denis and Erin.

- Alice chooses C, but tenant (Carol) hasn't chosen a house yet.
- Carol chooses E, then Alice chooses C.
- Bob chooses D, but tenant (Denis) hasn't chosen a house yet.

New order: Carol, Alice, Denis, Bob and Erin.

- Denis chooses D. A trivial cycle, so Denis is assigned D.

Endowment $\rightarrow$	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Carol, Alice, Denis, Bob and Erin.

- Bob chooses B. A trivial cycle, so Bob is assigned B.
- Erin chooses A. Tenant (Alice) has chosen a house.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = E \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Endowment →	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Carol, Alice, Denis, Bob and Erin.

- Bob chooses B. A trivial cycle, so Bob is assigned B.
- Erin chooses A. Tenant (Alice) has chosen a house.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = E \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

Endowment $\rightarrow$	A	B	C	D	
	P_{Alice}	P_{Bob}	P_{Carol}	P_{Denis}	P_{Erin}
	C	D	E	C	D
	D	E	C	E	E
	E	B	D	D	C
	A	C	B	B	A
	B	A	A	A	B

Order of dictators: Carol, Alice, Denis, Bob and Erin.

- Bob chooses B. A trivial cycle, so Bob is assigned B.
- Erin chooses A. Tenant (Alice) has chosen a house.
- The algorithm stops, final allocation:

$$\mu(\text{Alice}) = C \quad \mu(\text{Bob}) = B \quad \mu(\text{Carol}) = E \quad \mu(\text{Denis}) = D$$
$$\mu(\text{Erin}) = A$$

With YGMH—IGYT changing the order of dictators can yield to endless loops.

Endowment $\rightarrow$	A	B
	P_{Alice}	P_{Bob}
	B	A
	A	B

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

$$\text{Endowment} \rightarrow \begin{array}{cc} A & B \\ \hline P_{\text{Alice}} & P_{\text{Bob}} \\ \hline B & A \\ A & B \end{array}$$

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

Endowment $\rightarrow$

A	B
P_{Alice}	P_{Bob}
B	A
A	B

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

Endowment $\rightarrow$	A	B
	P_{Alice}	P_{Bob}
	B	A
	A	B

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop. The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

Endowment $\rightarrow$	A	B
	P_{Alice}	P_{Bob}
	B	A
	A	B

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

Endowment $\rightarrow$	A	B
	P_{Alice}	P_{Bob}
	B	A
	A	B

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

$$\text{Endowment} \rightarrow \begin{array}{cc} A & B \\ \hline P_{\text{Alice}} & P_{\text{Bob}} \\ \hline B & A \\ A & B \end{array}$$

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

$$\text{Endowment} \rightarrow \begin{array}{cc} A & B \\ \hline P_{\text{Alice}} & P_{\text{Bob}} \\ \hline B & A \\ A & B \end{array}$$

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

$$\text{Endowment} \rightarrow \begin{array}{cc} A & B \\ \hline P_{\text{Alice}} & P_{\text{Bob}} \\ \hline B & A \\ A & B \end{array}$$

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop. The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

With YGMH—IGYT changing the order of dictators can yield to endless loops.

$$\text{Endowment} \rightarrow \begin{array}{cc} A & B \\ \hline P_{\text{Alice}} & P_{\text{Bob}} \\ \hline B & A \\ A & B \end{array}$$

- Order is: Alice, Bob.
- Alice chooses B $\Rightarrow$ order changed to: Bob, Alice
- Bob chooses A $\Rightarrow$ order changed to: Alice, Bob
- repeat, ad nauseam.

Alice and Bob create a loop.The YGMH—IGYT algorithm states that in this case we do like TTC:

- Alice takes B
- Bob takes A

Top Trading Cycle with mixed endowments

This algorithm works like TTC when the objects have a priority:

- Individuals point to the house they want.
- Houses point to an individual.

To which individual will the house point?

- Draw a random order of the individuals.

  This order will become will become the priority ordering over individuals of all vacant houses.

- Occupied house point to their tenants.

Then run TTC.

Top Trading Cycle with mixed endowments

This algorithm works like TTC when the objects have a priority:

- Individuals point to the house they want.
- Houses point to an individual.

To which individual will the house point?

- Draw a random order of the individuals.

  This order will become will become the priority ordering over individuals of all vacant houses.

- Occupied house point to their tenants.

Then run TTC.

Top Trading Cycle with mixed endowments

This algorithm works like TTC when the objects have a priority:

- Individuals point to the house they want.
- Houses point to an individual.

To which individual will the house point?

- Draw a random order of the individuals.

  This order will become will become the priority ordering over individuals of all vacant houses.

- Occupied house point to their tenants.

Then run TTC.

Top Trading Cycle with mixed endowments

This algorithm works like TTC when the objects have a priority:

- Individuals point to the house they want.
- Houses point to an individual.

To which individual will the house point?

- Draw a random order of the individuals.

  This order will become will become the priority ordering over individuals of all vacant houses.

- Occupied house point to their tenants.

Then run TTC.

Top Trading Cycle with mixed endowments

This algorithm works like TTC when the objects have a priority:

- Individuals point to the house they want.
- Houses point to an individual.

To which individual will the house point?

- Draw a random order of the individuals.

  This order will become will become the priority ordering over individuals of all vacant houses.

- Occupied house point to their tenants.

Then run TTC.

Top Trading Cycle with mixed endowments

This algorithm works like TTC when the objects have a priority:

- Individuals point to the house they want.
- Houses point to an individual.

To which individual will the house point?

- Draw a random order of the individuals.

  This order will become will become the priority ordering over individuals of all vacant houses.

- Occupied house point to their tenants.

Then run TTC.

Results

Theorem

An assignment mechanism that uses the Top Trading Cycle algorithm with mixed endowment is (for any random ordering):

- *Efficient.*
- *Individually rational.*
- *Strategyproof*

Theorem

For any ordering of the individuals the YRMH–IGYT algorithm yields the same outcome as the Top Trading Cycle algorithm with mixed endowments (using the same ordering of the individuals).

Results

Theorem

An assignment mechanism that uses the Top Trading Cycle algorithm with mixed endowment is (for any random ordering):

- *Efficient.*
- *Individually rational.*
- *Strategyproof*

Theorem

For any ordering of the individuals the YRMH–IGYT algorithm yields the same outcome as the Top Trading Cycle algorithm with mixed endowments (using the same ordering of the individuals).

Results

Theorem
An assignment mechanism that uses the Top Trading Cycle algorithm with mixed endowment is (for any random ordering):

- *Efficient.*
- *Individually rational.*
- *Strategyproof*

Theorem
For any ordering of the individuals the YRMH–IGYT algorithm yields the same outcome as the Top Trading Cycle algorithm with mixed endowments (using the same ordering of the individuals).

Take-away

- In assignment models we have to match individuals to objects.
- Individuals have preferences over objects, but objects do not have preferences.

  $\Rightarrow$ Only individuals' welfare matter.
- Two cases: public endowments & private endowments.

  There is also the mix public-private endowment model.
- Serial dictatorship is a simple algorithm that is efficient and makes a mechanism using it strategyproof.

- The Top Trading Cycle algorithm is another solution that gives efficient assignments and ensures strategyproofness.

  TTC needs either:

  - private endowments; or
  - each object has a priority orderings over the individuals.

- Serial dictatorship does not ensures individually rational assignments when there are private endowments.

- With private endowments, individual rationality, efficiency and strategyproofness can be obtain with either:

  - the You Get My House — I Get Your Turn algorithm; or
  - the Top Trading Cycle with Mixed Endowments algorithm.

  Both algorithms yield the same outcome.