University of Glasgow

Summer School on Matching Problems, Markets and Mechanisms

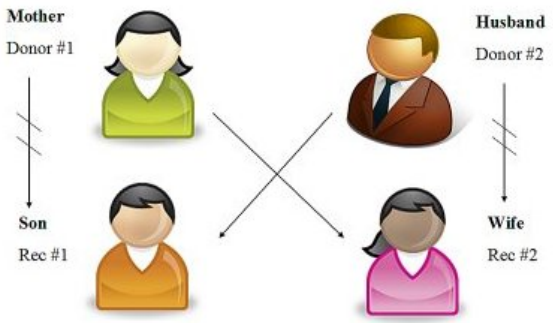
David Manlove

Outline

1. The Hospitals / Residents problem and its variants

2. The House Allocation problem

3. Kidney exchange



The House Allocation Problem

with applications to reviewer assignment

aka bipartite matching with one-sided preferences

House Allocation problem (HA)

- Set of applicants $A=\{a_1, a_2, \ldots, a_r\}$
- Set of houses $H=\{h_1, h_2, \ldots, h_s\}$

- Each applicant a_i has an *acceptable* set of houses $A_i \subseteq H$
- a_i ranks A_i in strict order of preference
- Houses do not have preferences over applicants

- Example:

$a_1 : h_2 \; h_1$
$a_2 : h_3 \; h_4 \; h_2$
$a_3 : h_4 \; h_3$
$a_4 : h_1 \; h_4$

a_1 finds h_1 and h_2 acceptable

a_3 prefers h_4 to h_3

- Let $n=r+s$ and let m=total length of preference lists
- "Applicants" could be reviewers and "houses" could be papers

Applications of HA

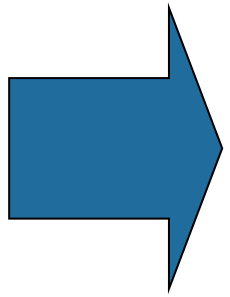
- Campus housing allocation in the US and Israel
  - Carnegie-Mellon, Duke, Michigan, Northwestern, Technion
  - [Chen and Sönmez, 2002; Perach, Polak and Rothblum, 2008]
- Allocating families to government-subsidised housing in China
  - [Yuan, 1996]
- Allocating students to projects and elective courses
  - Schools of Computing Science and Medicine, University of Glasgow
- DVD rental by post
  - [Abraham, Chen, Kumar and Mirrokni, 2006]
- Conference paper reviewer assignment (Easychair etc.)
  - [Garg, Kavitha, Kumar, Mehlhorn and Mestre, 2010]

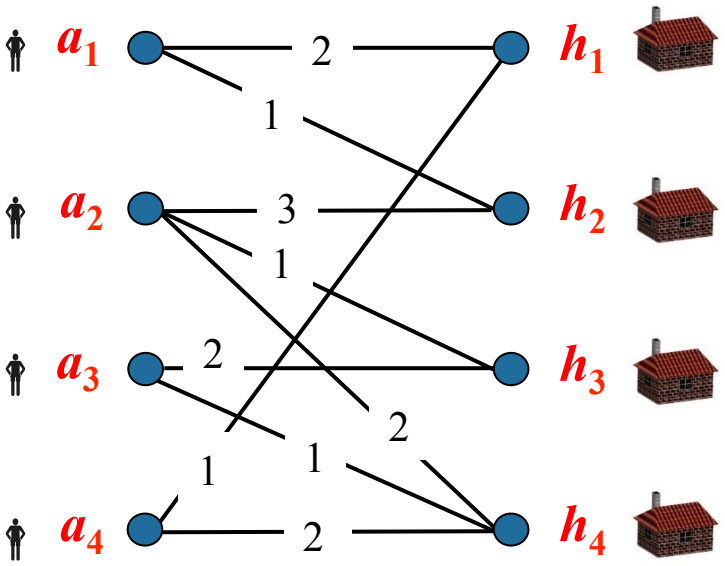
The underlying graph

- Weighted bipartite graph $G=(V,E)$
  - Vertex set $V=A \cup H$
  - Edge set: $\{a_i, h_j\} \in E$ if and only if a_i finds h_j acceptable
  - Weight of edge $\{a_i, h_j\}$ is rank of h_j in a_i's preference list

- Example:

  $a_1 : h_2\ h_1$
  $a_2 : h_3\ h_4\ h_2$
  $a_3 : h_4\ h_3$
  $a_4 : h_1\ h_4$





Definition of a matching

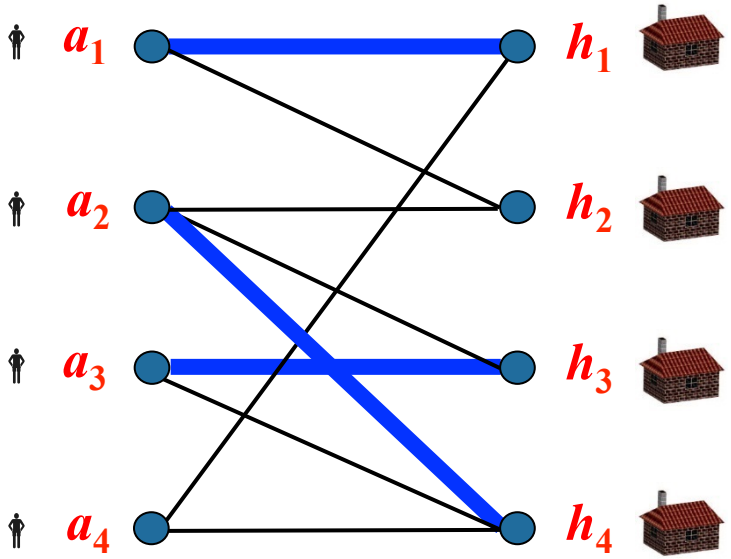
- A *matching* M is a subset of $A \times H$ such that:
  - if $(a_i, h_j) \in M$ then a_i finds h_j acceptable
  - each applicant belongs to at most one pair in M
  - each house belongs to at most one pair in M

- Example:

$a_1 : h_2\ h_1$
$a_2 : h_3\ h_4\ h_2$
$a_3 : h_4\ h_3$
$a_4 : h_1\ h_4$

$M(a_1)=h_1$

$M=\{(a_1, h_1), (a_2, h_4), (a_3, h_3)\}$



Maximum matchings

- In many applications we wish to match as many applicants as possible
- A *maximum matching* is a matching with largest size
- Example

$a_1 : h_2\ \ h_1$
$a_2 : h_3\ \ h_4\ \ h_2$
$a_3 : h_4\ \ h_3$
$a_4 : h_1\ \ h_4$

$a_1 : h_2\ \ h_1$
$a_2 : h_3\ \ h_4\ \ h_2$
$a_3 : h_4\ \ h_3$
$a_4 : h_1\ \ h_4$

- Some maximum matchings are “better” than others!
- Applicant a *prefers* matching M to matching M' if
  - a is matched in M but not in M', or
  - a is matched in both M and M' and prefers $M(a)$ to $M'(a)$

University of Glasgow

Tutorial Outline

2.1: Pareto optimal matchings

2.2: Popular matchings

2.3: Profile-based optimal matchings

2.1: Pareto optimal matchings

2.2: Popular matchings

2.3: Profile-based optimal matchings

Pareto optimal matchings

- A matching M_1 is *Pareto optimal* if there is no matching M_2 such that:
  1. Some applicant prefers M_2 to M_1
  2. No applicant prefers M_1 to M_2

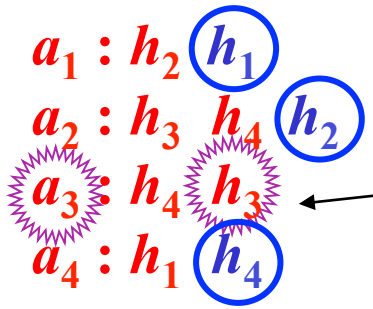
- Example

- M_1 is not Pareto optimal since a_1 and a_2 could swap houses – each would be better off!

- M_2 is Pareto optimal

Testing for Pareto optimality

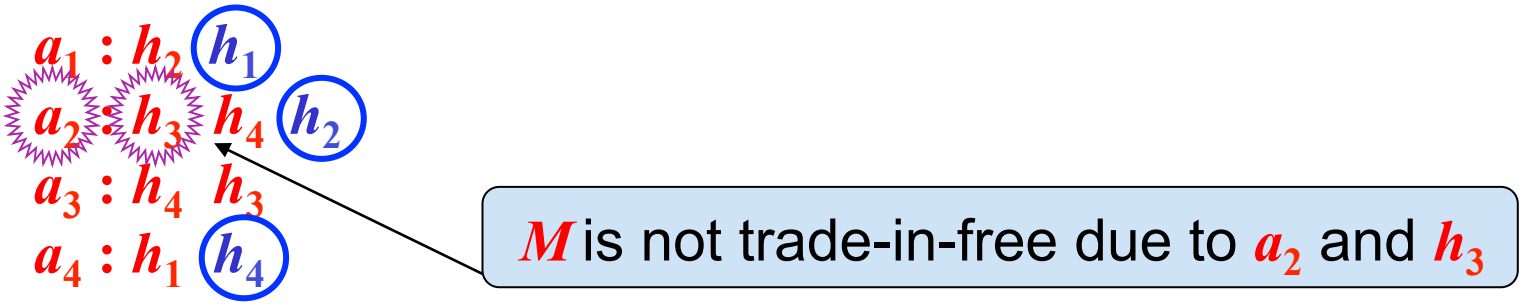
- A matching M is *maximal* if there is no applicant a and house h, each unmatched in M, such that a finds h acceptable



M is not maximal due to a_3 and h_3

Testing for Pareto optimality

- A matching M is *trade-in-free* if there is no matched applicant a and unmatched house h such that a prefers h to $M(a)$

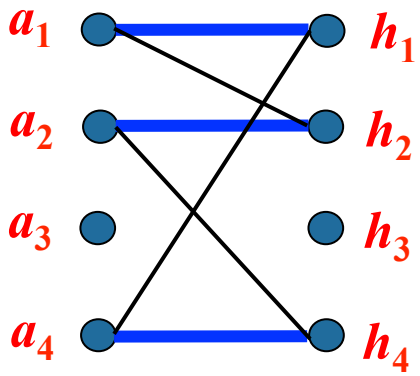


Testing for Pareto optimality

- A matching M is *coalition-free* if there is no *coalition*, i.e. a sequence of matched applicants $\langle a_0, a_1, \ldots, a_{r-1} \rangle$ such that a_i prefers $M(a_{i+1})$ to $M(a_i)$ $(0 \leq i \leq r\text{-}1)$

$a_1 : \underline{h_2}\ (h_1)$
$a_2 : h_3\ \underline{h_4}\ (h_2)$
$a_3 : h_4\ h_3$
$a_4 : \underline{h_1}\ (h_4)$

M is not coalition-free due to $\langle a_1, a_2, a_4 \rangle$



Testing for Pareto optimality

- A matching M is *maximal* if there is no applicant a and house h, each unmatched in M, such that a finds h acceptable
- A matching M is *trade-in-free* if there is no matched applicant a and unmatched house h such that a prefers h to $M(a)$
- A matching M is *coalition-free* if there is no *coalition*, i.e. a sequence of matched applicants $\langle a_0, a_1, \ldots, a_{r-1} \rangle$ such that a_i prefers $M(a_{i+1})$ to $M(a_i)$ $(0 \leq i \leq r\text{-}1)$

$a_1 : h_2$ (h_1)
$a_2 : h_3$ h_4 (h_2)
$a_3 : h_4$ h_3
$a_4 : h_1$ (h_4)

- Proposition: M is Pareto optimal if and only if M is maximal, trade-in-free and coalition-free

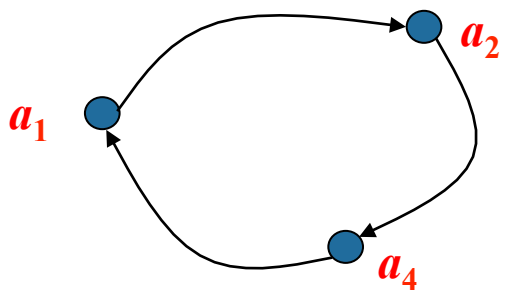
The envy graph

- Straightforward to check a given matching M for the maximality and trade-in-free properties in $O(m)$ time

- To check for the existence of a coalition:
  - Form the envy graph of M, denoted by $G(M)$
  - Vertex for each matched applicant
  - Edge from a_i to a_j if and only if a_i prefers $M(a_j)$ to $M(a_i)$

- Example

$a_1 : \underline{h_2}\ (h_1)$
$a_2 : h_3\ \underline{h_4}\ (h_2)$
$a_3 : h_4\ h_3$
$a_4 : \underline{h_1}\ (h_4)$



- M admits a coalition if and only if $G(M)$ has a directed cycle

- Proposition: we may check whether a given matching M is Pareto optimal in $O(m)$ time

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a “serial dictatorship mechanism” by economists

```
for each applicant a in turn
    if a has an unmatched house on his list
        match a to the most-preferred such house;
    else
        report a as unmatched;
```

- Example

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a "serial dictatorship mechanism" by economists

```
for each applicant a in turn
    if a has an unmatched house on his list
        match a to the most-preferred such house;
    else
        report a as unmatched;
```

- Example

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a “serial dictatorship mechanism” by economists

```
for each applicant a in turn
    if a has an unmatched house on his list
        match a to the most-preferred such house;
    else
        report a as unmatched;
```

- Example

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a "serial dictatorship mechanism" by economists

```
for each applicant a in turn
    if a has an unmatched house on his list
        match a to the most-preferred such house;
    else
        report a as unmatched;
```

- Example

a_1 : h_1 h_2 h_3
a_2 : h_1 h_2
a_3 : h_1 h_2

$M_1=\{(a_1,h_1), (a_2,h_2)\}$

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a "serial dictatorship mechanism" by economists

```
for each applicant a in turn
    if a has an unmatched house on his list
        match a to the most-preferred such house;
    else
        report a as unmatched;
```

- Example

a_1 : h_1 h_2 h_3
a_2 : h_1 h_2
a_3 : h_1 h_2

$M_1=\{(a_1,h_1), (a_2,h_2)\}$

a_1 : h_1 h_2 h_3
a_2 : h_1 h_2
a_3 : h_1 h_2

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a "serial dictatorship mechanism" by economists

```
for each applicant a in turn
   if a has an unmatched house on his list
      match a to the most-preferred such house;
   else
      report a as unmatched;
```

- Example

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

$M_1=\{(a_1,h_1), (a_2,h_2)\}$

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

Finding a Pareto optimal matching

- Simple greedy algorithm, called Algorithm Greedy-POM
  - referred to as a "serial dictatorship mechanism" by economists

```
for each applicant a in turn
    if a has an unmatched house on his list
        match a to the most-preferred such house;
    else
        report a as unmatched;
```

- Example

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

$M_1=\{(a_1,h_1), (a_2,h_2)\}$

$M_2=\{(a_1,h_3), (a_2,h_2), (a_3,h_1)\}$

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2$
$a_3 : h_1\ h_2$

- Proposition: Algorithm Greedy-POM constructs a Pareto optimal matching in $O(m)$ time

University of Glasgow

Finding a maximum Pareto optimal matching

- Take underlying weighted bipartite graph G
  - The weight of a matching M in G is the sum of the weights of the edges contained in M

$$wt(M) = \sum_{a \in A_M} rank_a(M(a))$$

    - where A_M is the set of applicants who are matched in M
    - $rank_a(h)$ is the rank of h in a's preference list
  - Find a maximum cardinality minimum weight matching M in G
  - M is Pareto optimal
  - For if not there exists a matching M' such that
    - $rank_a(M'(a)) \leq rank_a(M(a))$ for all $a \in A_M$
    - $rank_a(M'(a)) < rank_a(M(a))$ for some $a \in A_M$
  - so M' contradicts the minimum weight property of M
- A maximum cardinality minimum weight matching may be found in $O(\sqrt{n}m \log n)$ time
  - [Gabow and Tarjan, 1989]

University of Glasgow

A faster algorithm

- Three-phase algorithm with $O(\sqrt{nm})$ overall complexity
  - [Abraham, Cechlárová, M and Mehlhorn, 2004]
- Phase 1 – $O(\sqrt{nm})$ time
  - Find a maximum matching in G
  - Classical $O(\sqrt{nm})$ augmenting path algorithm
    - [Hopcroft and Karp, 1973]
- Phase 2 – $O(m)$ time
  - Enforce trade-in-free property
- Phase 3 – $O(m)$ time
  - Enforce coalition-free property
  - Extension of Gale's Top-Trading Cycles (TTC) algorithm
    - [Shapley and Scarf, 1974]

Phase 1

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$

$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$

$a_4 : h_3\ h_5\ h_4$

$a_5 : h_4\ h_3\ h_5$

$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ h_9$

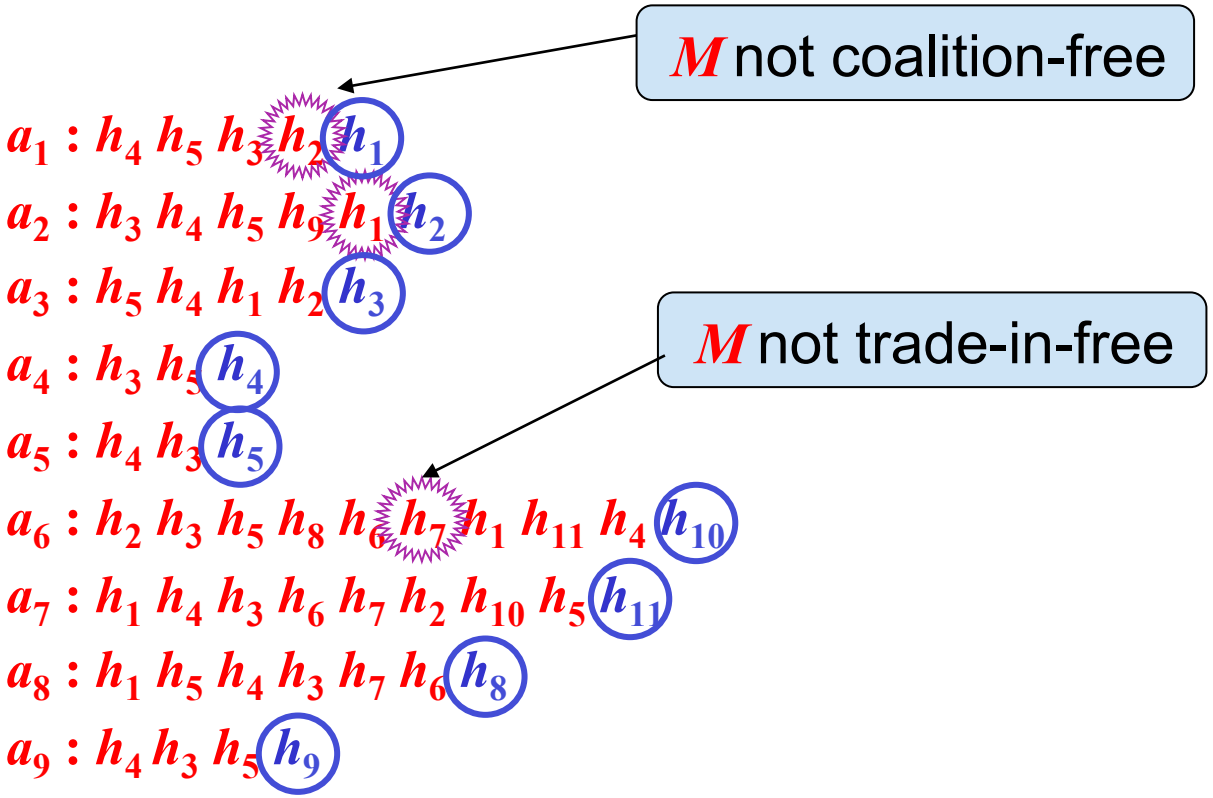
University of Glasgow

Phase 1

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ (h_{10})$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ (h_{11})$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

- Maximum matching M in G has size 9
- M must be maximal
- No guarantee that M is trade-in-free or coalition-free

Phase 1



- Maximum matching M in G has size 9
- M must be maximal
- No guarantee that M is trade-in-free or coalition-free

Phase 2 description

- For each house h, maintain a list L_h, initially containing those pairs (a, r) such that:
  - a is a matched applicant who prefers h to $M(a)$
  - r is the rank of h in a's list
- Maintain a stack S of unmatched houses h whose list L_h is nonempty
- Each matched applicant a maintains a pointer $curr_a$ to the rank of $M(a)$

- Example

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ (h_{10})$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ (h_{11})$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_6, 6), (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle (a_7, 7) \rangle$
$L_{h_{11}} = \langle (a_6, 8) \rangle$
$S = \langle h_6, h_7 \rangle$
$curr_{a_6} = 10$
$curr_{a_7} = 9$

Phase 2 description

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := r;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ (h_{10})$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ (h_{11})$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_6, 6), (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle (a_7, 7) \rangle$
$L_{h_{11}} = \langle (a_6, 8) \rangle$
$S = \langle h_6, h_7 \rangle$
$curr_{a_6} = 10$
$curr_{a_7} = 9$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ (h_{10})$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ (h_{11})$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_6, 6), (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle (a_7, 7) \rangle$
$L_{h_{11}} = \langle (a_6, 8) \rangle$
$S = \langle h_6, h_7 \rangle$
$curr_{a_6} = 10$
$curr_{a_7} = 9$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ (h_7)\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ (h_{11})$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle (a_7, 7) \rangle$
$L_{h_{11}} = \langle (a_6, 8) \rangle$
$S = \langle h_6, h_{10} \rangle$
$curr_{a_6} = 6$
$curr_{a_7} = 9$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ (h_7)\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ (h_{11})$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle (a_7, 7) \rangle$
$L_{h_{11}} = \langle (a_6, 8) \rangle$
$S = \langle h_6, h_{10} \rangle$
$curr_{a_6} = 6$
$curr_{a_7} = 9$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ (h_7)\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ (h_{10})\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle\ \rangle$
$L_{h_{11}} = \langle (a_6, 8) \rangle$
$S = \langle h_6, h_{11} \rangle$
$curr_{a_6} = 6$
$curr_{a_7} = 7$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ (h_7)\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ (h_{10})\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6}=\langle(a_6, 5), (a_7, 4), (a_8, 6)\rangle$
$L_{h_7}=\langle(a_7, 5), (a_8, 5)\rangle$
$L_{h_8}=\langle(a_6, 4)\rangle$
$L_{h_{10}}=\langle\ \rangle$
$L_{h_{11}}=\langle(a_6, 8)\rangle$
$S=\langle h_6, h_{11}\rangle$
$curr_{a_6}=6$
$curr_{a_7}=7$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ (h_7)\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ (h_{10})\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_6, 5), (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle\ \rangle$
$L_{h_{11}} = \langle\ \rangle$
$S = \langle h_6 \rangle$
$curr_{a_6} = 6$
$curr_{a_7} = 7$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ (h_{10})\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle\ \rangle$
$L_{h_{11}} = \langle\ \rangle$
$S = \langle h_7 \rangle$
$curr_{a_6} = 5$
$curr_{a_7} = 7$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ (h_{10})\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_7, 5), (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle\ \rangle$
$L_{h_{11}} = \langle\ \rangle$
$S = \langle h_7 \rangle$
$curr_{a_6} = 5$
$curr_{a_7} = 7$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle\ \rangle$
$L_{h_{11}} = \langle\ \rangle$
$S = \langle\ \rangle$
$curr_{a_6} = 5$
$curr_{a_7} = 5$

Phase 2 example

```
while S is nonempty
    pop a house h from S;
    remove the first pair (a,r) from the head of L_h;
    if r < curr_a // a prefers h to M(a)
        let h' = M(a);
        remove (a,h') from M and add (a,h) to M;
        curr_a := h;
        h := h';
    push h onto the stack if L_h is nonempty;
```

$a_1 : h_4\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : h_5\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : h_3\ h_5\ (h_4)$
$a_5 : h_4\ h_3\ (h_5)$
$a_6 : h_2\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

$L_{h_6} = \langle (a_7, 4), (a_8, 6) \rangle$
$L_{h_7} = \langle (a_8, 5) \rangle$
$L_{h_8} = \langle (a_6, 4) \rangle$
$L_{h_{10}} = \langle\ \rangle$
$L_{h_{11}} = \langle\ \rangle$
$S = \langle\ \rangle$
$curr_{a_6} = 5$
$curr_{a_7} = 5$

University of Glasgow

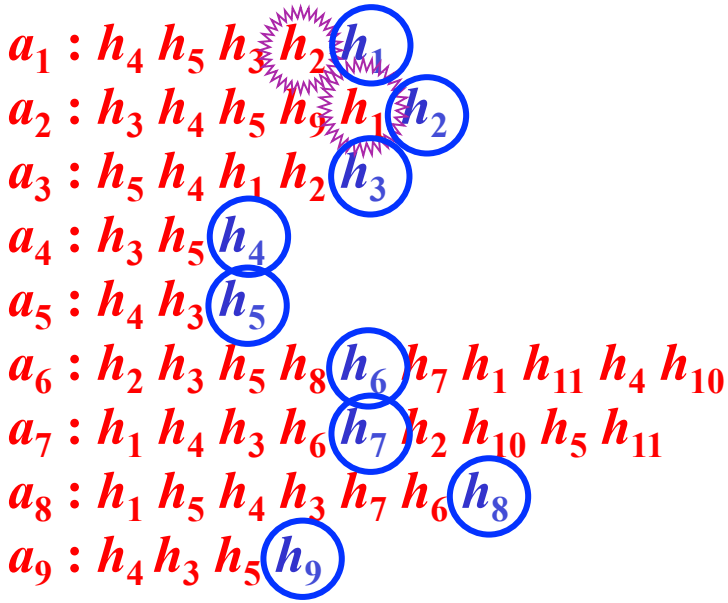
Phase 2 termination

- Once Phase 2 terminates, matching is trade-in-free
- Data structures may be initialised in $O(m)$ time
- Main loop takes $O(m)$ time overall
- ∴ Phase 2 is $O(m)$
- Coalitions may remain…

Phase 2 termination

- Once Phase 2 terminates, matching is trade-in-free
- Data structures may be initialised in $O(m)$ time
- Main loop takes $O(m)$ time overall
- ∴ Phase 2 is $O(m)$
- Coalitions may remain…

- Example



University of Glasgow

Phase 3 description

- Once Phase 2 terminates, matching is trade-in-free
- Still need to eliminate potential coalitions
- Repeatedly finding and eliminating coalitions takes $O(m^2)$ time
  - Cycle detection in $G(M)$ takes $O(m)$ time
  - $O(m)$ coalitions in the worst case
- Faster method: extension of TTC algorithm
  - An applicant matched to his/her first-choice house cannot be in a coalition
    - Such an applicant can be removed from consideration
  - Houses matched to such applicants are no longer exchangeable
    - Such a house can be removed from consideration
  - This rule can be recursively applied until either
    - No applicant remains (matching is coalition-free)
    - A coalition exists, which can be found and removed

University of Glasgow

Phase 3 description

- Build a path P of applicants (represented by a stack)
- Each house is initially unlabelled
- Each applicant a has a pointer $p(a)$ pointing to $M(a)$ or the first unlabelled house on a's preference list (whichever comes first)
- Keep a counter $c(a)$ for each applicant a (initially $c(a)=0$)
  - This represents the number of times a appears on the stack
- Outer loop iterates over each matched applicant a such that $p(a) \neq M(a)$
- Initialise P to contain applicant a
- Inner loop iterates while P is nonempty
  - Pop an applicant a' from P
  - If $c(a')=2$ we have a coalition (CYCLE)
    - Remove by popping the stack and label the houses involved
  - Else if $p(a')=M(a')$ we reach a dead end (BACKTRACK)
    - Label $M(a')$
  - Else add a'' where $p(a')=M(a'')$ to the path (EXTEND)
    - Push a' and a'' onto the stack
    - Increment $c(a'')$

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : \underline{h_3}\ h_5\ (h_4)$
$a_5 : \underline{h_4}\ h_3\ (h_5)$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	

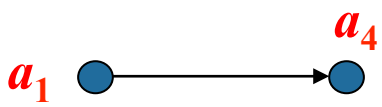
a_1 

$P=\langle a_1 \rangle$

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : \underline{h_3}\ h_5\ (h_4)$
$a_5 : \underline{h_4}\ h_3\ (h_5)$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	1	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



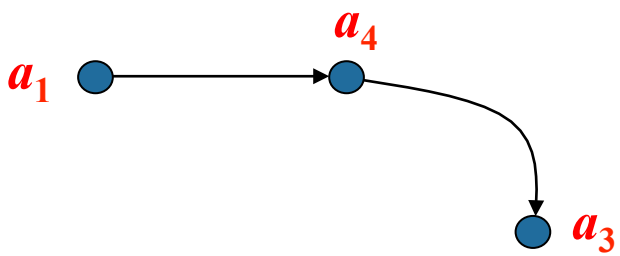
EXTEND

$P = \langle a_1, a_4 \rangle$

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : \underline{h_3}\ h_5\ (h_4)$
$a_5 : \underline{h_4}\ h_3\ (h_5)$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	1	h_3	
a_4	1	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



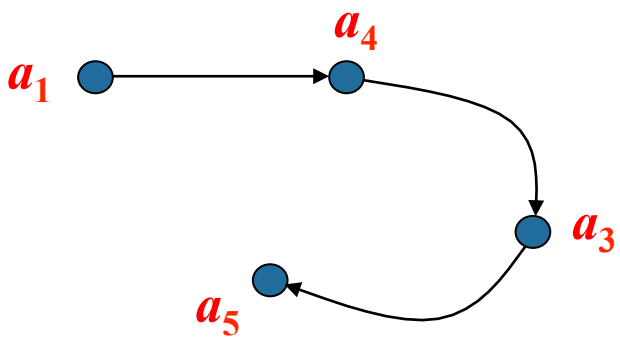
EXTEND

$P = \langle a_1, a_4, a_3 \rangle$

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : \underline{h_3}\ h_5\ (h_4)$
$a_5 : \underline{h_4}\ h_3\ (h_5)$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	1	h_3	
a_4	1	h_4	
a_5	1	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



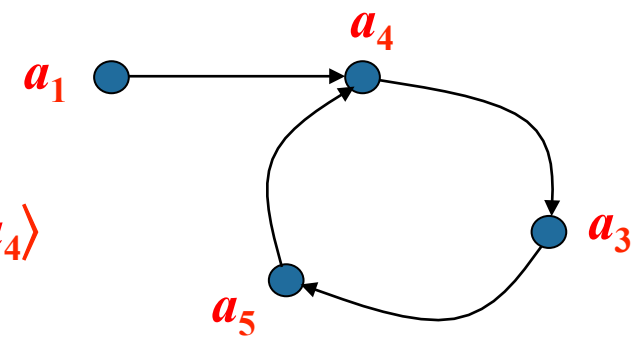
$P = \langle a_1, a_4, a_3, a_5 \rangle$

EXTEND

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : \underline{h_3}\ h_5\ (h_4)$
$a_5 : \underline{h_4}\ h_3\ (h_5)$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	1	h_3	
a_4	2	h_4	
a_5	1	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



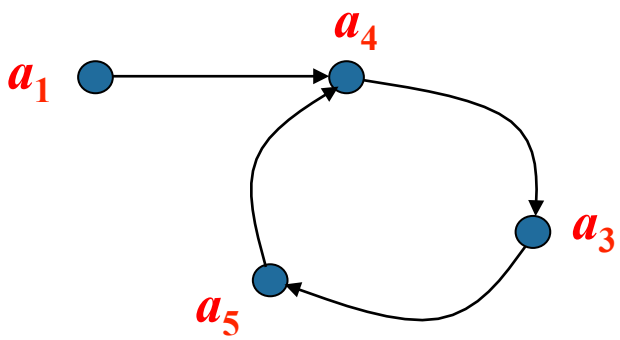
$P = \langle a_1, a_4, a_3, a_5, a_4 \rangle$

EXTEND

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ (h_3)$
$a_4 : \underline{h_3}\ h_5\ (h_4)$
$a_5 : \underline{h_4}\ h_3\ (h_5)$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	1	h_3	
a_4	2	h_4	
a_5	1	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



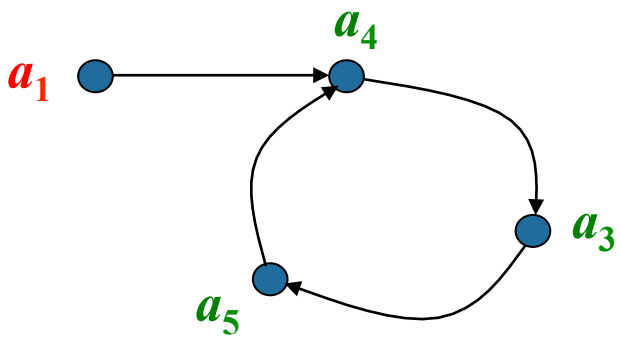
$P = \langle a_1, a_4, a_3, a_5 \rangle$

CYCLE

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : (\underline{h_5})\ h_4\ h_1\ h_2\ h_3$
$a_4 : (\underline{h_3})\ h_5\ h_4$
$a_5 : (\underline{h_4})\ h_3\ h_5$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



$P = \langle a_1, a_4, a_3, a_5 \rangle$

CYCLE

Phase 3: example

$a_1 : \underline{h_4}\ h_5\ h_3\ h_2\ (h_1)$
$a_2 : \underline{h_3}\ h_4\ h_5\ h_9\ h_1\ (h_2)$
$a_3 : (\underline{h_5})\ h_4\ h_1\ h_2\ h_3$
$a_4 : (\underline{h_3})\ h_5\ h_4$
$a_5 : (\underline{h_4})\ h_3\ h_5$
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ (h_6)\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ (h_7)\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ (h_8)$
$a_9 : \underline{h_4}\ h_3\ h_5\ (h_9)$

applicant	counter	house	label
a_1	1	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	

a_1 

$P=\langle a_1 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$
$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$
$a_4 : h_3\ h_5\ h_4$
$a_5 : h_4\ h_3\ h_5$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$
$a_9 : h_4\ h_3\ h_5\ h_9$

applicant	counter	house	label
a_1	1	h_1	
a_2	1	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	

EXTEND

$P=\langle a_1, a_2 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$

$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$

$a_4 : h_3\ h_5\ h_4$

$a_5 : h_4\ h_3\ h_5$

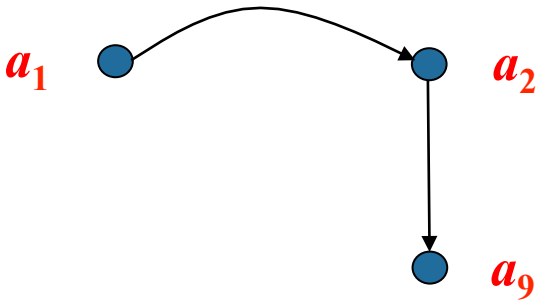
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ h_9$

applicant	counter	house	label
a_1	1	h_1	
a_2	1	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	1	h_9	



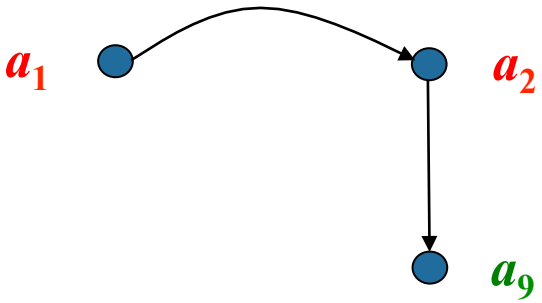
EXTEND

$P = \langle a_1, a_2, a_9 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$
$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$
$a_4 : h_3\ h_5\ h_4$
$a_5 : h_4\ h_3\ h_5$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$
$a_9 : h_4\ h_3\ h_5\ h_9$

applicant	counter	house	label
a_1	1	h_1	
a_2	1	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	1	h_9	



$P=\langle a_1, a_2 \rangle$

BACKTRACK

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$
$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$
$a_4 : h_3\ h_5\ h_4$
$a_5 : h_4\ h_3\ h_5$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$
$a_9 : h_4\ h_3\ h_5\ h_9$

applicant	counter	house	label
a_1	1	h_1	
a_2	1	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	

$P=\langle a_1, a_2 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$

$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$

$a_4 : h_3\ h_5\ h_4$

$a_5 : h_4\ h_3\ h_5$

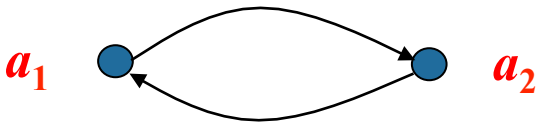
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ h_9$

applicant	counter	house	label
a_1	2	h_1	
a_2	1	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



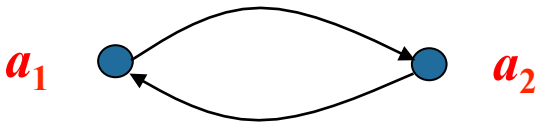
EXTEND

$P = \langle a_1, a_2, a_1 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ h_2\ h_1$
$a_2 : h_3\ h_4\ h_5\ h_9\ h_1\ h_2$
$a_3 : h_5\ h_4\ h_1\ h_2\ h_3$
$a_4 : h_3\ h_5\ h_4$
$a_5 : h_4\ h_3\ h_5$
$a_6 : h_2\ h_3\ h_5\ h_8\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$
$a_9 : h_4\ h_3\ h_5\ h_9$

applicant	counter	house	label
a_1	2	h_1	
a_2	1	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



CYCLE

$P=\langle a_1, a_2 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

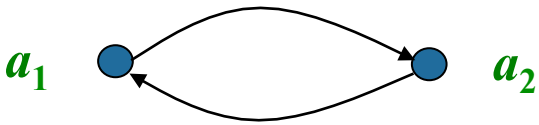
$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ \underline{h_6}\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ \underline{h_7}\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ \underline{h_8}$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



CYCLE

$P = \langle a_1, a_2 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

$a_6 : \underline{h_2}\ h_3\ h_5\ h_8\ \underline{h_6}\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ \underline{h_7}\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ \underline{h_8}$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	1	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	

a_6

$P=\langle a_6 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

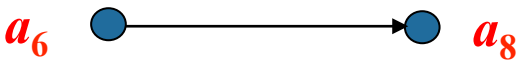
$a_6 : h_2\ h_3\ h_5\ \underline{h_8}\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : \underline{h_1}\ h_5\ h_4\ h_3\ h_7\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	1	h_6	
a_7	0	h_7	
a_8	1	h_8	
a_9	0	h_9	



EXTEND

$P = \langle a_6, a_8 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

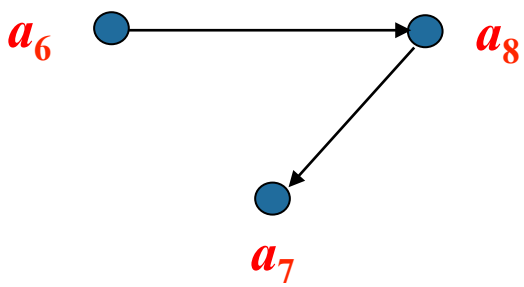
$a_6 : h_2\ h_3\ h_5\ \underline{h_8}\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : \underline{h_1}\ h_4\ h_3\ h_6\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ \underline{h_7}\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	1	h_6	
a_7	1	h_7	
a_8	1	h_8	
a_9	0	h_9	



$P = \langle a_6, a_8, a_7 \rangle$

EXTEND

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

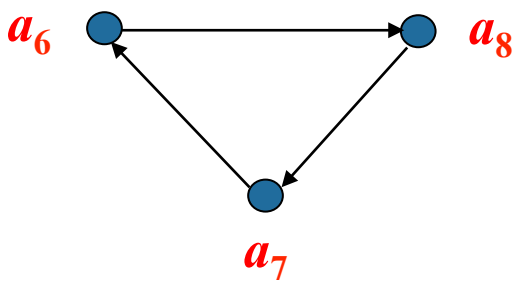
$a_6 : h_2\ h_3\ h_5\ \underline{h_8}\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ \underline{h_6}\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ \underline{h_7}\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	2	h_6	
a_7	1	h_7	
a_8	1	h_8	
a_9	0	h_9	



$P = \langle a_6, a_8, a_7, a_6 \rangle$

EXTEND

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

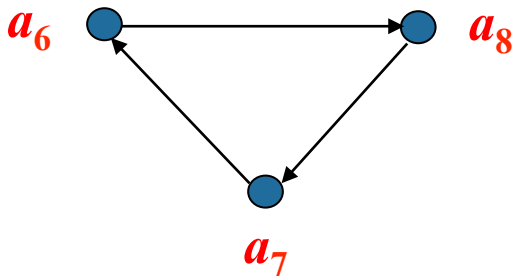
$a_6 : h_2\ h_3\ h_5\ \underline{h_8}\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ \underline{h_6}\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ \underline{h_7}\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	2	h_6	
a_7	1	h_7	
a_8	1	h_8	
a_9	0	h_9	



$P = \langle a_6, a_8, a_7 \rangle$

CYCLE

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

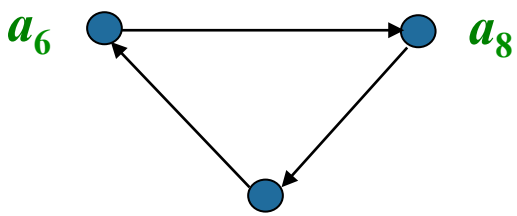
$a_6 : h_2\ h_3\ h_5\ \underline{h_8}\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ \underline{h_6}\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ \underline{h_7}\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	



CYCLE

$P = \langle a_6, a_8, a_7 \rangle$

Phase 3: example

$a_1 : h_4\ h_5\ h_3\ \underline{h_2}\ h_1$

$a_2 : h_3\ h_4\ h_5\ h_9\ \underline{h_1}\ h_2$

$a_3 : \underline{h_5}\ h_4\ h_1\ h_2\ h_3$

$a_4 : \underline{h_3}\ h_5\ h_4$

$a_5 : \underline{h_4}\ h_3\ h_5$

$a_6 : h_2\ h_3\ h_5\ \underline{h_8}\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$

$a_7 : h_1\ h_4\ h_3\ \underline{h_6}\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$

$a_8 : h_1\ h_5\ h_4\ h_3\ \underline{h_7}\ h_6\ h_8$

$a_9 : h_4\ h_3\ h_5\ \underline{h_9}$

applicant	counter	house	label
a_1	0	h_1	
a_2	0	h_2	
a_3	0	h_3	
a_4	0	h_4	
a_5	0	h_5	
a_6	0	h_6	
a_7	0	h_7	
a_8	0	h_8	
a_9	0	h_9	

$P = \langle\rangle$

University of Glasgow

Phase 3 termination

- Once Phase 3 terminates, matching is coalition-free
- Data structures may be initialised in $O(m)$ time
- Nested loops take $O(m)$ time overall
- ∴Phase 3 is $O(m)$

$a_1 : h_4\ h_5\ h_3\ (h_2)\ h_1$
$a_2 : h_3\ h_4\ h_5\ h_9\ (h_1)\ h_2$
$a_3 : (h_5)\ h_4\ h_1\ h_2\ h_3$
$a_4 : (h_3)\ h_5\ h_4$
$a_5 : (h_4)\ h_3\ h_5$
$a_6 : h_2\ h_3\ h_5\ (h_8)\ h_6\ h_7\ h_1\ h_{11}\ h_4\ h_{10}$
$a_7 : h_1\ h_4\ h_3\ (h_6)\ h_7\ h_2\ h_{10}\ h_5\ h_{11}$
$a_8 : h_1\ h_5\ h_4\ h_3\ (h_7)\ h_6\ h_8$
$a_9 : h_4\ h_3\ h_5\ (h_9)$

- **Theorem:** given an instance of HA, a maximum Pareto optimal matching can be found in $O(\sqrt{n}m)$ time.

University of Glasgow

Housing markets

- Each applicant owns a house initially – their *initial endowment* M_0
- Any matching M must be *individually rational*
  - for each applicant a, either $M(a)=M_0(a)$ or a prefers M to M_0
- An individually rational matching M is in the *strict core* if there is no other matching M' that *weakly blocks* with respect to the *coalition* S
  - the members of S can only improve by exchanging their own resources
  - some member of S prefers M' to M
  - no member of S prefers M to M'
- Every Housing Market admits a unique strict core matching (using the TTC algorithm)
  - [Roth and Postlewaite, 1977]
  - [Shapley and Scarf, 1974]
- The TTC algorithm is strategy-proof
  - [Roth, 1982a]

2.1: Pareto optimal matchings

2.2: Popular matchings

2.3: Profile-based optimal matchings

University of Glasgow

Popular matchings

- A matching M is *popular* if there is no matching M' such that more applicants prefer M' to M than prefer M to M'
- Define the relation $\leftarrow$ on matchings by $M \leftarrow M'$ if more applicants prefer M to M' than prefer M' to M
- A popular matching is a minimal element in the relation $\leftarrow$
- A popular matching need not exist, e.g.,

$$a_1 : h_1 \quad h_2 \quad h_3$$
$$a_2 : h_1 \quad h_2 \quad h_3$$
$$a_3 : h_1 \quad h_2 \quad h_3$$

Popular matchings may not exist

- A matching M is *popular* if there is no matching M' such that more applicants prefer M' to M than prefer M to M'
- Define the relation $\leftarrow$ on matchings by $M \leftarrow M'$ if more applicants prefer M to M' than prefer M' to M
- A popular matching is a minimal element in the relation $\leftarrow$
- A popular matching need not exist, e.g.,

$$a_1 : (h_1)\ h_2\ h_3$$
$$a_2 : h_1\ (h_2)\ h_3$$
$$a_3 : h_1\ h_2\ (h_3)$$

- The blue matching shown is unique up to symmetry

Popular matchings may not exist

- A matching M is *popular* if there is no matching M' such that more applicants prefer M' to M than prefer M to M'
- Define the relation $\leftarrow$ on matchings by $M \leftarrow M'$ if more applicants prefer M to M' than prefer M' to M (i.e., M' is *more popular* than M)
- A popular matching is a minimal element in the relation $\leftarrow$
- A popular matching need not exist, e.g.,

$a_1 : h_1\ h_2\ h_3$
$a_2 : h_1\ h_2\ h_3$
$a_3 : h_1\ h_2\ h_3$

- The blue matching shown is unique up to symmetry
- The green matching M' satisfies $M' \leftarrow M$

University of Glasgow

Popular matchings can have different sizes

$a_1 : h_1 \quad h_4$

$a_2 : h_2 \quad h_5$

$a_3 : h_3 \quad h_4 \quad h_6$

$a_4 : h_1$

$a_5 : h_2$

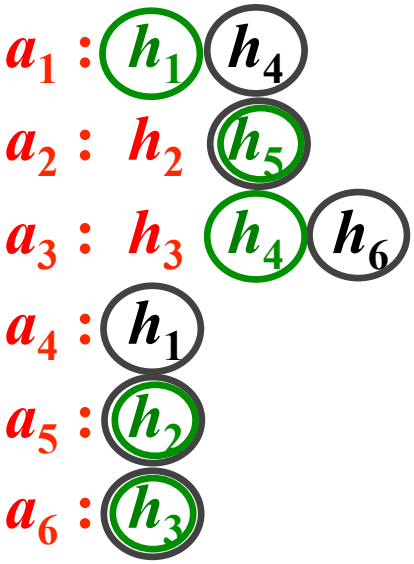
$a_6 : h_3$

Popular matchings can have different sizes

$a_1 : h_1$ (h_4)
$a_2 : h_2$ (h_5)
$a_3 : h_3$ h_4 (h_6)
$a_4 : (h_1)$
$a_5 : (h_2)$
$a_6 : (h_3)$

- The black matching is the unique maximum matching

Popular matchings can have different sizes



- The black matching is the unique maximum matching
- It isn't popular: the green matching M' satisfies $M' \leftarrow M$

Popular matchings can have different sizes

$a_1 : (h_1)\ h_4$
$a_2 :\ h_2\ (h_5)$
$a_3 :\ h_3\ (h_4)\ h_6$
$a_4 :\ h_1$
$a_5 : (h_2)$
$a_6 : (h_3)$

- In fact the green matching is itself popular (size **5**)

Popular matchings can have different sizes

$a_1 : h_1 \; h_4$

$a_2 : h_2 \; h_5$

$a_3 : h_3 \; h_4 \; h_6$

$a_4 : h_1$

$a_5 : h_2$

$a_6 : h_3$

- In fact the green matching is itself popular (size **5**)
- So is the blue matching (size **3**)

Popular matchings can have different sizes

$a_1 : (h_1)\ h_4$
$a_2 :\ h_2\ (h_5)$
$a_3 :\ h_3\ (h_4)\ h_6$
$a_4 :\ h_1$
$a_5 : (h_2)$
$a_6 : (h_3)$

- There is no popular matching of size **6**
- So the green matching is a maximum popular matching

University of Glasgow

Summary of results

- Is there a polynomial-time algorithm to determine whether a popular matching exists, and if so to find one?
- Problem introduced by [Gärdenfors, 1975] in the context of two-sided preferences
- Neat (and surprising) characterisation of popular matchings
- Case of strict preferences:
  - $O(n+m)$ algorithm to determine whether a popular matching exists, and if so to find a largest one
- Ties in the preference lists:
  - $O(\sqrt{n}m)$ algorithm for the same problem
- [Abraham, Irving, Kavitha, Mehlhorn, 2005]

Graph representation

- Bipartite graph G, applicant vertices $A = \{a_1, \ldots, a_r\}$, house vertices $H = \{h_1, \ldots, h_s\}$
- Edge $\{a_i, h_j\}$ of weight k if h_j is the k^{th} choice of a_i
- E_k is the set of all edges of weight k
- Example:

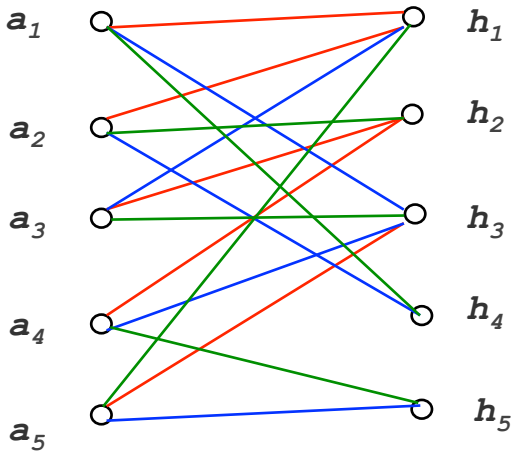
a_1: h_1 h_3 h_4

a_2: h_1 h_4 h_2

a_3: h_2 h_1 h_3

a_4: h_2 h_3 h_5

a_5: h_3 h_5 h_1



E_1 in red

E_2 in blue

E_3 in green

Last resorts

- For each applicant a append a unique house $l(a)$ to the end of his list – his *last resort*
  - This ensures every maximal matching is *applicant-complete*
  - The *size* of a matching is the number of applicants not matched to their last resort

Example:

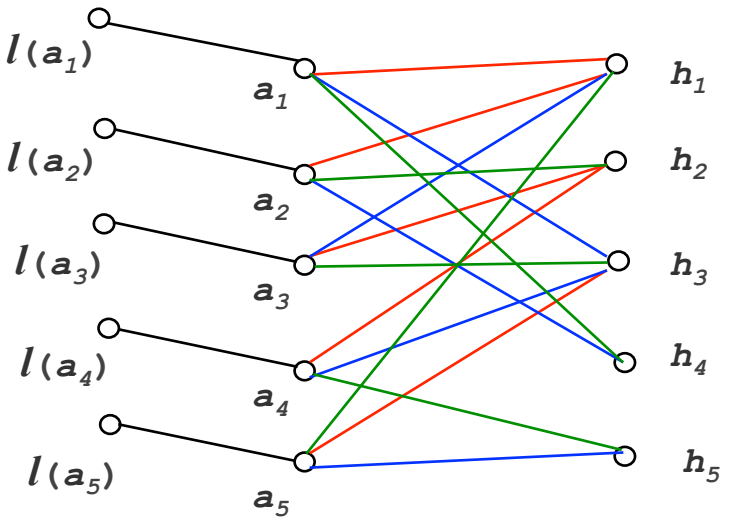
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



E_1 in red

E_2 in blue

E_3 in green

E_4 in black

University of Glasgow

f-houses and s-houses

- For applicant a, $f(a)$ denotes his first choice house
- House h is an ***f-house*** if $h = f(a)$ for some a
- For applicant a, $s(a)$ denotes the first non *f*-house on his list
  - $s(a)$ is bound to exist (because of $l(a)$)

University of Glasgow

f-houses and s-houses

- For applicant a, $f(a)$ denotes his first choice house
- House h is an ***f-house*** if $h = f(a)$ for some a
- For applicant a, $s(a)$ denotes the first non *f*-house on his list
  - $s(a)$ is bound to exist (because of $l(a)$)

Example:

a_1: h_1 h_3 h_4 $l(a_1)$
a_2: h_1 h_4 h_2 $l(a_2)$
a_3: h_2 h_1 h_3 $l(a_3)$
a_4: h_2 h_3 h_5 $l(a_4)$
a_5: h_3 h_5 h_1 $l(a_5)$

- f-houses in green – f-houses are h_1, h_2 and h_3

University of Glasgow

f-houses and s-houses

- For applicant a, $f(a)$ denotes his first choice house
- House h is an ***f-house*** if $h = f(a)$ for some a
- For applicant a, $s(a)$ denotes the first non *f*-house on his list
  - $s(a)$ is bound to exist (because of $l(a)$)

Example:

a_1: h_1 h_3 h_4 $l(a_1)$
a_2: h_1 h_4 h_2 $l(a_2)$
a_3: h_2 h_1 h_3 $l(a_3)$
a_4: h_2 h_3 h_5 $l(a_4)$
a_5: h_3 h_5 h_1 $l(a_5)$

- f-houses in green – f-houses are h_1, h_2 and h_3
- s-houses in blue – s-houses are $l(a_3)$, h_4 and h_5

University of Glasgow

Lemmas and Theorem

- **Lemma 1**: If M is a popular matching, and h is an f-house then $M(h) = a$ for some a such that $h = f(a)$.

- **Lemma 2**: If M is a popular matching, then $M(a)$ cannot lie between $f(a)$ and $s(a)$ on a's preference list.

- **Lemma 3**: If M is a popular matching, then $M(a)$ is never below $s(a)$ on a's preference list.

- **Theorem 1**: M is a popular matching if and only if

  (i) every f-house is matched in M, and

  (ii) for each applicant a, $M(a) = f(a)$ or $M(a) = s(a)$.

The reduced graph G'

- For each applicant vertex a_i, delete all incident edges except those to $f(a_i)$ and $s(a_i)$

Example

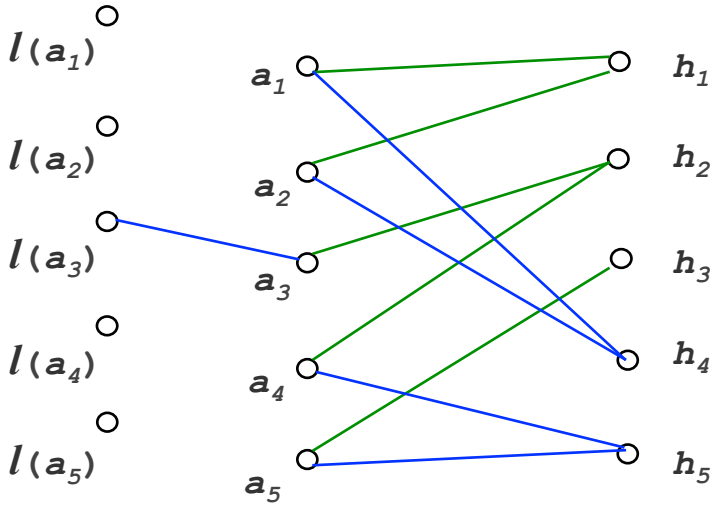
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



$f(a_i)$ in green

$s(a_i)$ in blue

$(a_i, f(a_i))$ in green

$(a_i, s(a_i))$ in blue

The reduced graph G'

- **Theorem 2**: Matching M is popular if and only if

  (i) every f-house is matched in M; and

  (ii) M is an applicant-complete matching in the reduced graph G'

Example

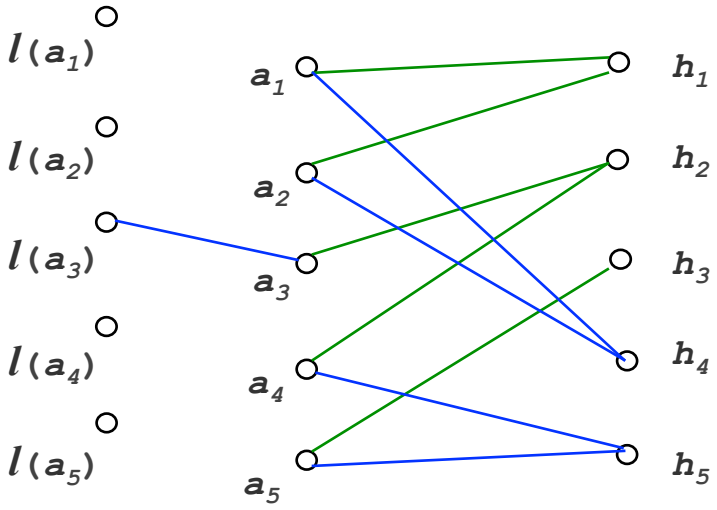
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



The reduced graph G'

- **Theorem 2**: Matching M is popular if and only if

  (i) every f-house is matched in M; and

  (ii) M is an applicant-complete matching in the reduced graph G'

Example

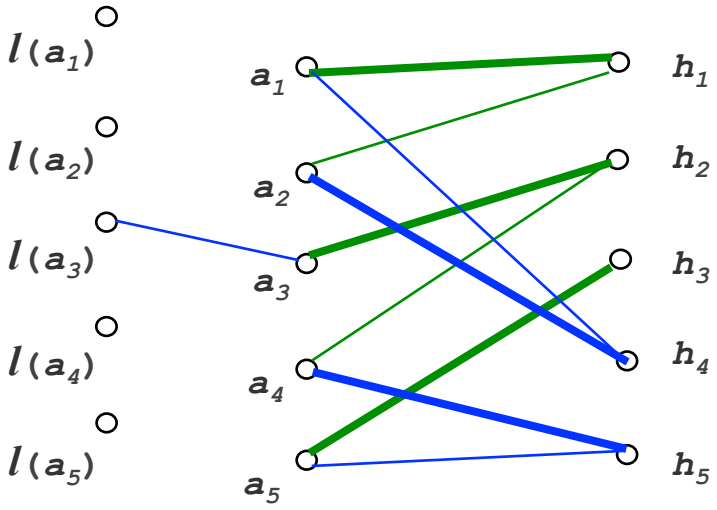
a_1 : h_1 h_3 h_4 $l(a_1)$

a_2 : h_1 h_4 h_2 $l(a_2)$

a_3 : h_2 h_1 h_3 $l(a_3)$

a_4 : h_2 h_3 h_5 $l(a_4)$

a_5 : h_3 h_5 h_1 $l(a_5)$



Popular matching algorithm

```
form G = (V = A ∪ H, E), the graph of the instance;
form G', the reduced graph of G;
if G' admits an applicant-complete matching M   (*)
   for each f-house h unmatched in M
      a = any applicant in f(h);
      promote a to h in M;
else
   no popular matching exists;
```

- Complexity:
  - all steps except (*) are clearly $O(m + n)$
    - $O(m)$ if $n = O(m)$
  - standard matching algorithm for (*) no better than $O(n^{3/2})$

Finding an applicant-complete matching in G'

```
M = ∅;
while some house h has degree 1
   a = unique applicant adjacent to h;
   M = M ∪ {(a,h)};
   G' = G' - {a, h};   // remove a and h from G'
while some house h has degree 0
   G' = G' - {h};
// every surviving house has degree at least 2
// every surviving applicant also has degree 2
if |surviving houses| < |surviving applicants|
   no applicant-complete matching exists;
else
   // G' is a family of disjoint cycles
   return M ∪ any maximum matching of G';
```

- all steps are $O(n + m)$

Example execution

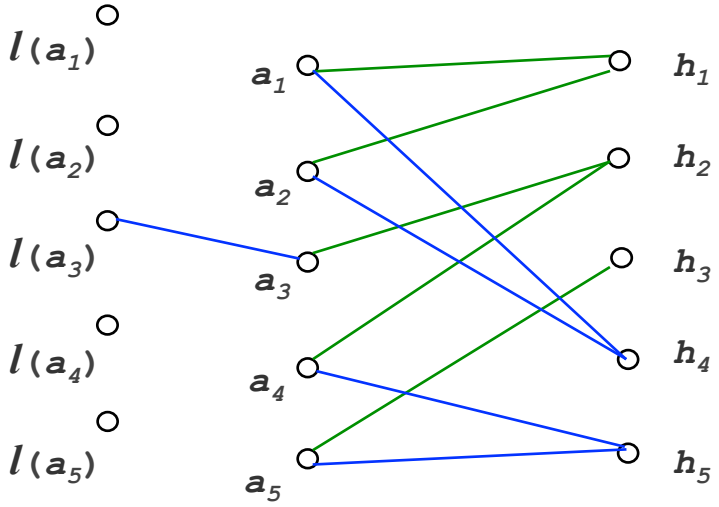
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \emptyset$
  - house $l(a_3)$ has degree 1; add $(a_3, l(a_3))$ to M
  - remove a_3 and $l(a_3)$ from G'

Example execution

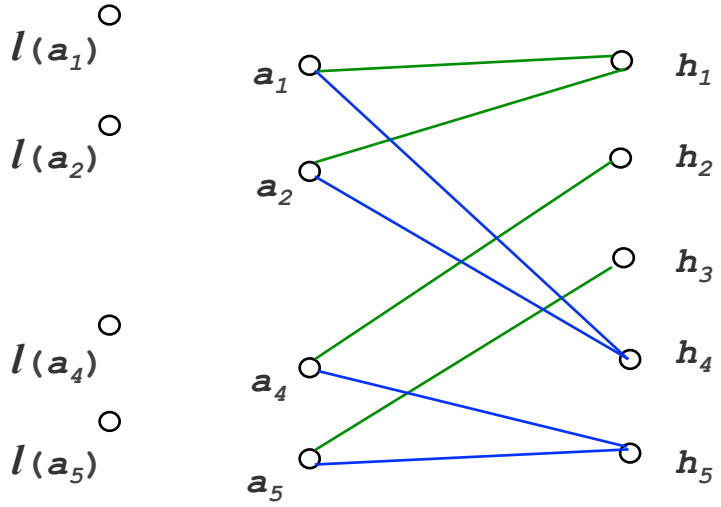
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_3, l(a_3))\}$
  - house h_2 has degree 1; add (a_4, h_2) to M
  - remove a_4 and h_2 from G'

Example execution

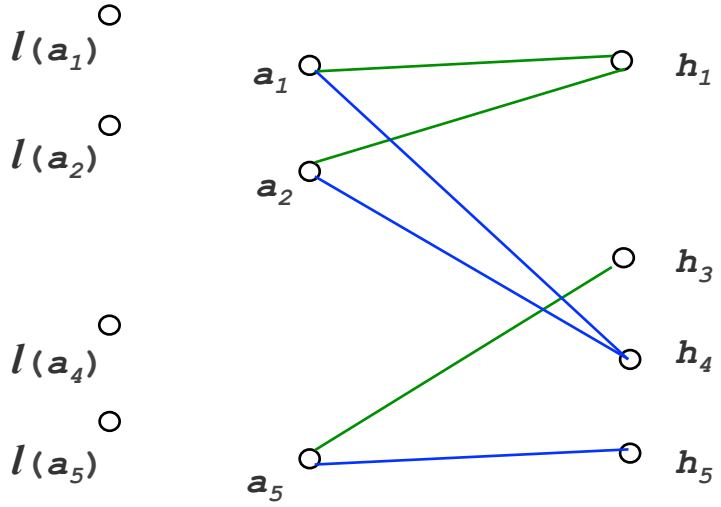
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_3, l(a_3)), (a_4, h_2)\}$
  - house h_5 has degree 1; add (a_5, h_5) to M
  - remove a_5 and h_5 from G'

Example execution

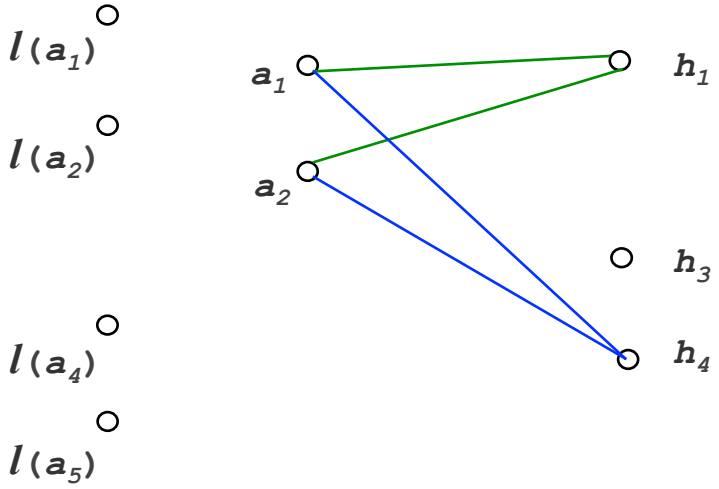
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_3, l(a_3)), (a_4, h_2), (a_5, h_5)\}$
  - remove degree zero houses h_3, $l(a_1)$, $l(a_2)$, $l(a_4)$, $l(a_5)$

Example execution

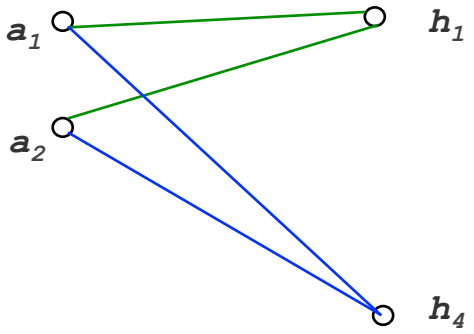
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_3, l(a_3)), (a_4, h_2), (a_5, h_5)\}$
  - no. of surviving houses = no. of surviving applicants
  - G' consists of just one cycle;
  - traverse the cycle adding alternate edges to M;

Example execution

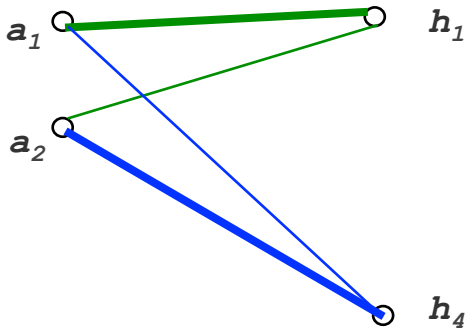
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_1, h_1), (a_2, h_4), (a_3, l(a_3)), (a_4, h_2), (a_5, h_5)\}$

Example execution

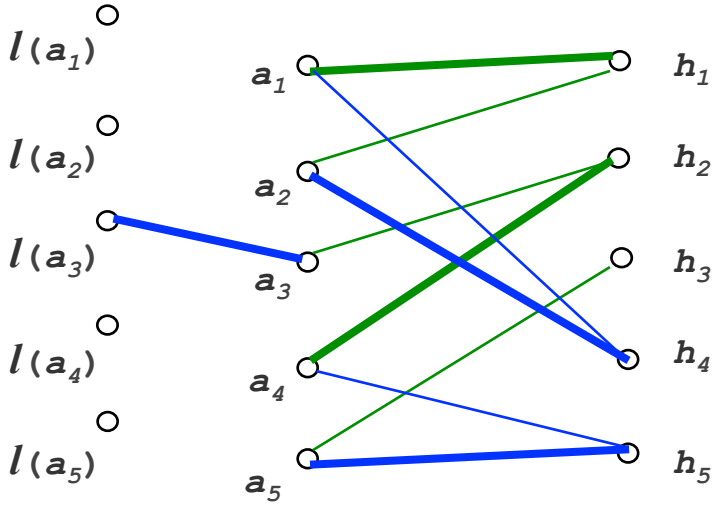
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_1, h_1), (a_2, h_4), (a_3, l(a_3)), (a_4, h_2), (a_5, h_5)\}$
  - h_3 is an unmatched f-house in M;
  - promote a_5 to h_3

Example execution

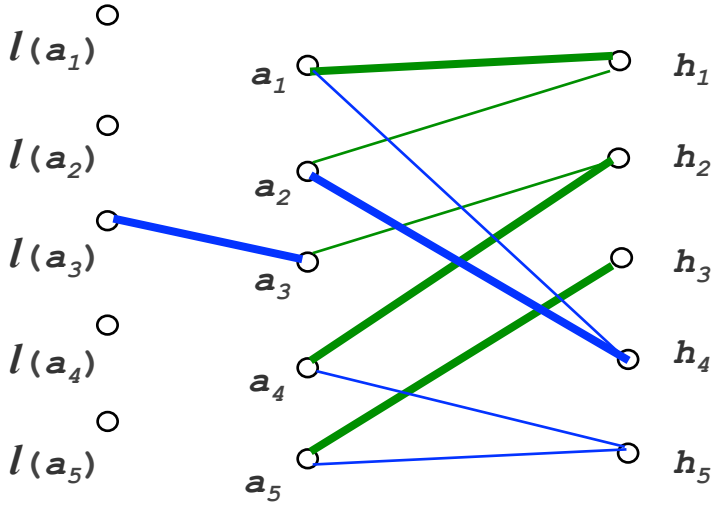
a_1: h_1 h_3 h_4 $l(a_1)$

a_2: h_1 h_4 h_2 $l(a_2)$

a_3: h_2 h_1 h_3 $l(a_3)$

a_4: h_2 h_3 h_5 $l(a_4)$

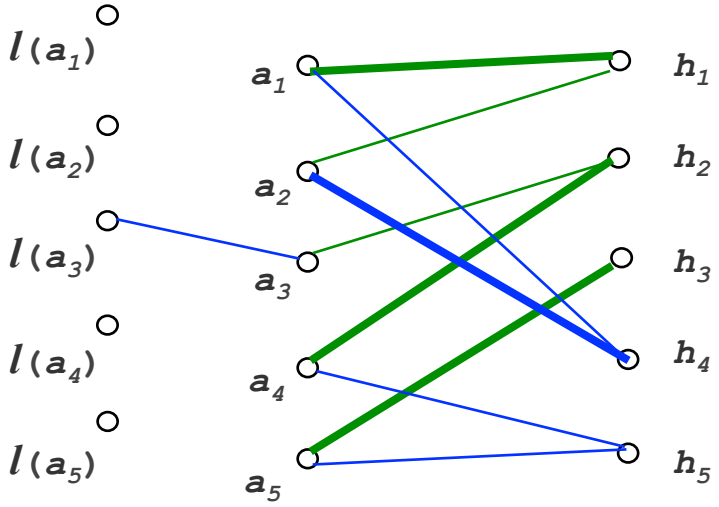
a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_1, h_1), (a_2, h_4), (a_3, l(a_3)), (a_4, h_2), (a_5, h_3)\}$
- Matching M with last resort edges removed is now popular

Example execution

a_1: h_1 h_3 h_4 $l(a_1)$
a_2: h_1 h_4 h_2 $l(a_2)$
a_3: h_2 h_1 h_3 $l(a_3)$
a_4: h_2 h_3 h_5 $l(a_4)$
a_5: h_3 h_5 h_1 $l(a_5)$



- $M = \{(a_1, h_1), (a_2, h_4), (a_4, h_2), (a_5, h_3)\}$
- Matching M is now popular
- **Theorem:** There is an $O(n+m)$ algorithm that finds a popular matching or reports that none exists, given an instance of HA.

- Straightforward extension to find a maximum popular matching
- Non-trivial extension to the case of ties in the preference lists ($O(\sqrt{n}m)$ time)
  - [Abraham, Irving, Kavitha and Mehlhorn, 2005]

Tutorial Outline

2.1: Pareto optimal matchings

2.2: Popular matchings

2.3: Profile-based optimal matchings

The profile of a matching

- The *degree* of a matching M is the maximum k such that some applicant has their kth-choice house in M
- Example:

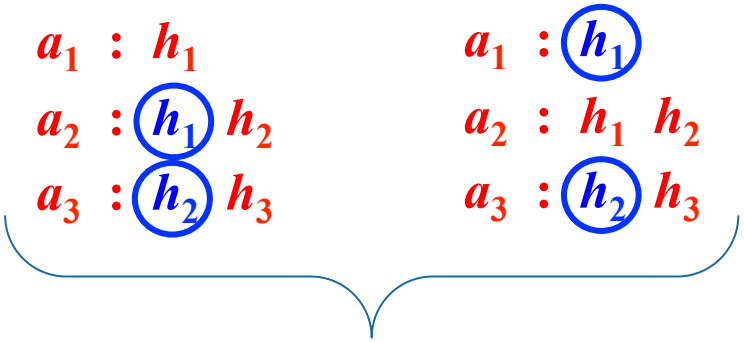
a_1:	**(h_1)**	h_3	h_4
a_2:	h_1	**(h_4)**	h_2
a_3:	h_2	h_1	**(h_3)**
a_4:	**(h_2)**	h_3	h_5
a_5:	h_3	**(h_5)**	h_1

  - $M = \{(a_1, h_1), (a_2, h_4), (a_3, h_3), (a_4, h_2), (a_5, h_5)\}$
  - Degree of M is 3
- The *profile* of M is a vector $\langle x_1, x_2, \ldots, x_k \rangle$ where k is the degree of M and x_i is the number of applicants who have their ith-choice house in M
  - Profile of M is $\langle 2, 2, 1 \rangle$

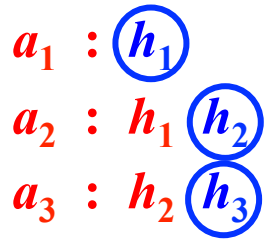
Rank-maximal matchings

- A matching M is *rank-maximal* if its profile is lexicographically maximum
  - So in M, the maximum number of applicants obtain their first-choice house, and subject to this, the maximum number obtain their second-choice house, etc.
  - A rank-maximal matching M can be computed in $O(\min\{r^*\sqrt{n},\ n+r^*\}m)$ time (where $n=|V|$, $m=|E|$, and r^* is the degree of M)
  - [Irving, Kavitha, Mehlhorn, Michail and Paluch, 2004]
- A rank-maximal matching need not be maximum!
- Example

$a_1 : h_1$
$a_2 : h_1\ h_2$
$a_3 : h_2\ h_3$



rank-maximal matchings – profile $\langle 2,0\rangle$



maximum matching – profile $\langle 1,2\rangle$

Other profile-based optimal matchings

- A matching M is *greedy maximum* [Irving, 2007] if
  1. M is maximum
  2. subject to 1, M has lexicographically maximum profile
- A matching M is *generous* [Irving, 2007] if
  1. M is maximum
  2. subject to 1, the reverse profile of M is lexicographically minimum

$a_1 : (h_1)\ h_2\ h_3\ h_4\ h_5$
$a_2 : h_1\ (h_2)\ h_3\ h_4\ h_5$
$a_3 : h_1\ h_2\ h_3\ (h_4)\ h_5$
$a_4 : h_1\ (h_3)\ h_5\ h_4\ h_2$
$a_5 : h_2\ (h_5)\ h_4\ h_3\ h_1$

- Profile $\langle 1,3,0,1 \rangle$
- Weight **11**
- Minimum weight maximum matching

$a_1 : (h_1)\ h_2\ h_3\ h_4\ h_5$
$a_2 : h_1\ h_2\ h_3\ (h_4)\ h_5$
$a_3 : h_1\ h_2\ h_3\ h_4\ (h_5)$
$a_4 : h_1\ (h_3)\ h_5\ h_4\ h_2$
$a_5 : (h_2)\ h_5\ h_4\ h_3\ h_1$

- Profile $\langle 2,1,0,1,1 \rangle$
- Weight **13**
- Greedy maximum matching

$a_1 : (h_1)\ h_2\ h_3\ h_4\ h_5$
$a_2 : h_1\ (h_2)\ h_3\ h_4\ h_5$
$a_3 : h_1\ h_2\ (h_3)\ h_4\ h_5$
$a_4 : h_1\ h_3\ (h_5)\ h_4\ h_2$
$a_5 : h_2\ h_5\ (h_4)\ h_3\ h_1$

- Profile $\langle 1,1,3 \rangle$
- Weight **12**
- Generous matching

University of Glasgow

Other profile-based optimal matchings

- A matching M is *greedy maximum* [Irving, 2007] if
  1. M is maximum
  2. subject to 1, M has lexicographically maximum profile
- A matching M is *generous* [Irving, 2007] if
  1. M is maximum
  2. subject to 1, the reverse profile of M is lexicographically minimum
- A matching M is *greedy generous* [Irving, 2007] if
  1. M is maximum
  2. the degree of M equals that of a generous matching
  3. subject to 1 and 2, M has lexicographically maximum profile
- A greedy maximum / generous / greedy generous matching M can be found in $O(r^* \sqrt{n} m \log n)$ time (where r^* is the degree of M, $n=|V|$ and $m=|E|$)
  - [Huang and Kavitha, 2012]

Additional constraints

- Reviewer assignment problem considered by [Garg et al, 2010]
- *Coverage*: there will be a value of $t \geq 1$ such each paper must be reviewed t times
- *Load balancing*: each reviewer should be given roughly the same number of papers
- *Fairness*:

$r_1 : (p_1\ p_2)\ (p_3\ p_4)$

$r_2 : (p_1\ p_2)\ (p_3\ p_4)$

Additional constraints

- Reviewer assignment problem considered by [Garg et al, 2010]
- *Coverage*: there will be a value of $t \geq 1$ such each paper must be reviewed t times
- *Load balancing*: each reviewer should be given roughly the same number of papers
- *Fairness*:

  Both matchings have profile $\langle 2,2 \rangle$ but clearly second is fairer

- Solution: introduce profile for each reviewer
  - $p(r_1,M_1)=\langle 2,0 \rangle$, $p(r_2,M_1)=\langle 0,2 \rangle$
  - $p(r_1,M_2)=\langle 1,1 \rangle$, $p(r_2,M_2)=\langle 1,1 \rangle$
- Create sorted vector of profiles (worst to best according to lexicographic order)
  - $\langle \langle 0,2 \rangle, \langle 2,0 \rangle \rangle$ for M_1
  - $\langle \langle 1,1 \rangle, \langle 1,1 \rangle \rangle$ for M_2

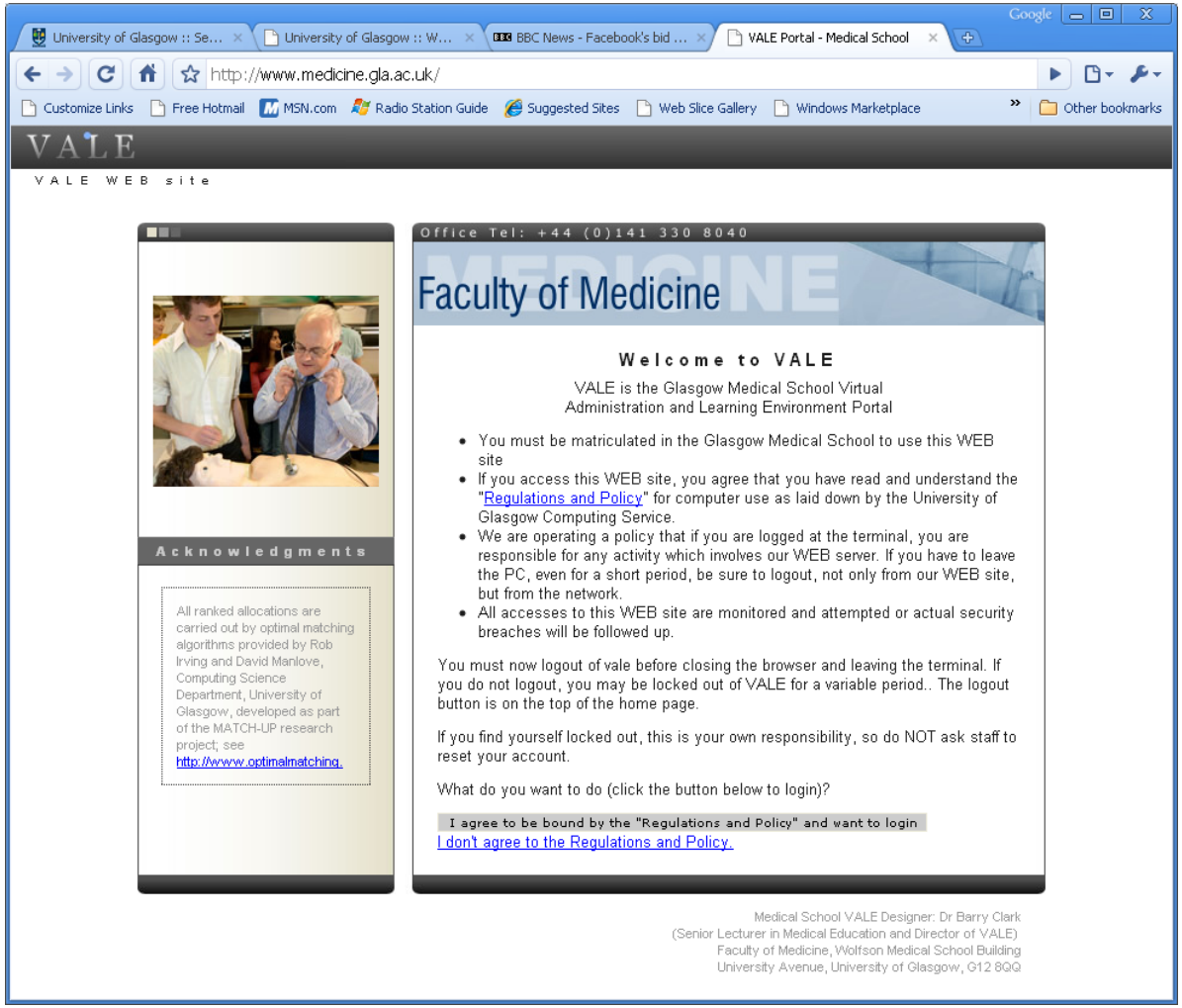
Leximin optimal matchings

- [Garg et al, 2010]: *leximin optimal matching*
  - lexicographic ordering on vectors of sorted profiles
  - $\langle\langle 1,1\rangle, \langle 1,1\rangle\rangle$ is better than $\langle\langle 0,2\rangle, \langle 2,0\rangle\rangle$
  - so M_2 is the leximin optimal matching

- Finding a leximin optimal matching is NP-hard in general, but solvable in polynomial time if maximum rank of a paper is **2** [Garg et al, 2010]

Course allocation at the School of Medicine

- VALE: Virtual Administration and Learning Environment
  - Handles various allocation tasks, including:
    - Allocating medical students to "SSCs" (elective courses)
    - Allocating medical students to short-term hospital placements



- **http://www.medicine.gla.ac.uk**

University of Glasgow

2006-07 data

- 246 medical students bidding for 17 SSCs, each with a fixed capacity

SSC #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Capacity	60	20	30	3	60	4	2	24	4	8	30	3	2	20	10	14	14

- Each preference list of length 6
- Outcome:

	Minimum cost	Greedy	Generous	Greedy generous
Size	246	246	246	246
Weight	463	521	479	478
Profile	⟨119,74,30,13,6,4⟩	⟨151,28,15,13,17,22⟩	⟨95,87,46,18⟩	⟨118,60,32,36⟩

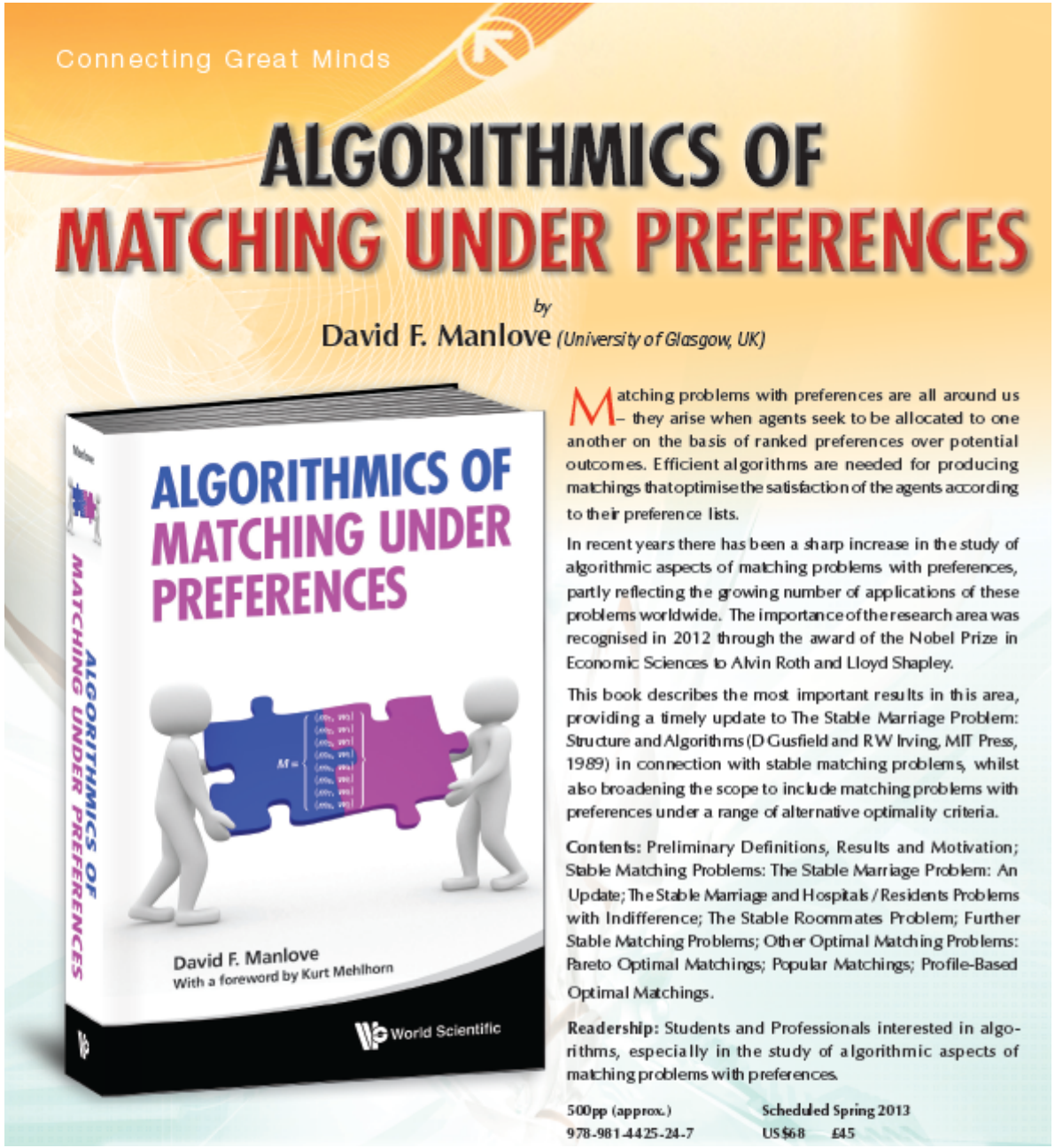
- Serial dictatorship mechanism produced matching with size 228, weight 478 and profile ⟨117,49,21,16,14,11⟩

University of Glasgow

Open problems

- Maximum Pareto optimal matching
  - Ties in the preference lists
    - Solvable in $O(\sqrt{n}m \log n)$ time. Can we achieve $O(\sqrt{n}m)$ time?
- Popular matchings in the Roommates problem
  - Single set of participants
  - Each participant has a (strict) preference list containing a subset of the others
  - Efficient algorithm to find a popular matching or report that none exists?
- Profile-based optimal matchings
  - Structure of the set of rank-maximal matchings
  - Similarly for other types of profile-based optimal matchings
  - Profile-based optimal matchings in the non-bipartite case
- Acknowledgements: Rob Irving; Baharak Rastegari

Further reading



- Chapters 6, 7, 8

University of Glasgow

References

Abraham, D.J., Blum, A. and Sandholm, T. (2007). Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges, in Proceedings of EC ’07: the 8th ACM Conference on Electronic Commerce (ACM), pp. 295–304

Abraham, D.J., Cechlárová, K., Manlove, D.F. and Mehlhorn, K. (2004). Pareto optimality in house allocation problems, in Proceedings of ISAAC ’04: the 15th Annual International Symposium on Algorithms and Computation, Lecture Notes in Computer Science, Vol. 3341 (Springer), pp. 3–15

Abraham, D.J., Chen, N., Kumar, V. and Mirrokni, V.S. (2006). Assignment problems in rental markets, in Proceedings of WINE ’06: the 2nd International Workshop on Internet and Network Economics, Lecture Notes in Computer Science, Vol. 4286 (Springer), pp. 198–213

Abraham, D.J., Irving, R.W., Kavitha, T. and Mehlhorn, K. (2005). Popular matchings, in Proceedings of SODA ’05: the 16th ACM-SIAM Symposium on Discrete Algorithms (ACM-SIAM), pp. 424–432

Ashlagi, I., Fischer, F., Kash, I. and Procaccia, A. D. (2010). Mix and match, in Proceedings of EC ’10: the 11th ACM Conference on Electronic Commerce (ACM), pp. 305–314

University of Glasgow

References

Ashlagi, I. and Roth, A. (2011). Individual rationality and participation in large scale, multi-hospital kidney exchange, in Proceedings of EC ’11: the 12th ACM Conference on Electronic Commerce (ACM), pp. 321–322

Ashlagi, I. and Roth, A. (2012). New challenges in multihospital kidney exchange, American Economic Review 102, 3, pp. 354–359

Askalidis, G., Immorlica, I., Kwanashie, A., Manlove, D.F., Pountourakis, E. (2013). Socially Stable matchings in the Hospitals / Residents problem. To appear in Proceedings of WADS 2013: the 13th Algorithms and Data Structures Symposium, Lecture Notes in Computer Science, Springer, 2013

Biró, P., Irving, R.W. and Schlotter, I. (2011). Stable matching with couples: an empirical study, ACM Journal of Experimental Algorithmics 16, section 1, article 2, 27 pages

Biró, P., Manlove, D.F. and Mittal, S. (2010). Size versus stability in the Marriage problem. Theoretical Computer Science 411, pp. 1828-1841

Biró, P., Manlove, D.F. and Rizzi, R (2009). Maximum weight cycle packing in directed graphs, with application to kidney exchange, Discrete Mathematics, Algorithms and Applications 1, 4, pp. 499–517

University of Glasgow

References

Caragiannis, I., Filos-Ratsikas, A. and Procaccia, A. (2011). An improved 2-agent kidney exchange mechanism, in Proceedings of WINE '11: the 7th International Workshop on Internet and Network Economics, Lecture Notes in Computer Science Series, vol. 7090 (Springer), pp. 37–48

Chen, Y. and Sönmez, T. (2002). Improving efficiency of on-campus housing: An experimental study, American Economic Review 92, 5, pp. 1669–1686

Conway, J.H. (1976). Personal communication, reported in Knuth, D.E. (1976). Mariages Stables (Les Presses de L'Université de Montréal). English translation in Stable Marriage and its Relation to Other Combinatorial Problems, volume 10 of CRM Proceedings and Lecture Notes, American Mathematical Society, 1997

Dubins, L.E. and Freedman, D.A. (1981). Machiavelli and the Gale-Shapley algorithm, American Mathematical Monthly 88, 7, pp. 485–494

Gabow, H.N. and Tarjan, R.E. (1989). Faster scaling algorithms for network problems, SIAM Journal on Computing 18, pp. 1013–1036

Gale, D. and Shapley, L.S. (1962). College admissions and the stability of marriage, American Mathematical Monthly 69, pp. 9–15

University of Glasgow

References

Gale, D. and Sotomayor, M. (1985). Ms. Machiavelli and the stable matching problem, American Mathematical Monthly 92, 4, pp. 261–268

Gale, D. and Sotomayor, M. (1985). Some remarks on the stable matching problem, Discrete Applied Mathematics 11, pp. 223–232

Gärdenfors, P (1975). Match making: assignments based on bilateral preferences, Behavioural Science 20, pp. 166–173

Garg, N., Kavitha, T., Kumar, A., Mehlhorn, K. and Mestre, J. (2010). Assigning papers to referees, Algorithmica 58, 1, pp. 119–136

Glorie, K.M., Klundert, J.J. van de and Wagelmans, A. (2013). Iterative branch-and-price for hierarchical multi-criteria kidney exchange. Econometric Institute Research Paper EI 2012-11, Erasmus University Rotterdam

Gusfield, D. and Irving, R.W. (1989). The Stable Marriage Problem: Structure and Algorithms (MIT Press)

Huang, C.-C. (2006). Cheating by men in the Gale-Shapley stable matching algorithm, in Proceedings of ESA ’06: the 14th Annual European Symposium on Algorithms, Lecture Notes in Computer Science, Vol. 4168 (Springer), pp. 418–431

University of Glasgow

References

Huang, C.-C. and Kavitha, T. (2012). Weight-maximal matchings, in Proceedings of MATCH-UP ’12: the 2nd International Workshop on Matching Under Preferences, pp. 87–98

Immorlica, N. and Mahdian, M. (2005). Marriage, honesty and stability, in Proceedings of SODA ’05: the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (ACM-SIAM), pp. 53–62

Irving, R.W. (1985). An efficient algorithm for the “stable roommates” problem, Journal of Algorithms, 6, pp. 577-595

Irving, R.W. (2007). Greedy and generous matchings via a variant of the Bellman-Ford algorithm, Unpublished manuscript

Irving, R.W., Kavitha, T., Mehlhorn, K., Michail, D. and Paluch, K. (2004). Rank-maximal matchings, in Proceedings of SODA ’04: the 15th ACM-SIAM Symposium on Discrete Algorithms (ACM-SIAM), pp. 68–75

Irving, R.W. and Manlove, D.F. (2009). Finding large stable matchings, ACM Journal of Experimental Algorithmics 14, section 1, article 2, 30 pages

University of Glasgow

References

Irving, R.W., Manlove, D.F. and O'Malley, G. (2009). Stable marriage with ties and bounded length preference lists, Journal of Discrete Algorithms 7, 2, pp. 213–219

Irving, R.W., Manlove, D.F. and Scott, S. (2008). The stable marriage problem with master preference lists, Discrete Applied Mathematics 156, 15, pp. 2959–2977

Iwama, K., Manlove, D., Miyazaki, S. and Morita, Y. (1999). Stable marriage with incomplete lists and ties, in Proceedings of ICALP '99: the 26th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science, Vol. 1644 (Springer), pp. 443–452

Keizer, K.M. , de Klerk, M., Haase-Kromwijk, B.J.J.M., and Weimar, W. (2005). The Dutch algorithm for allocation in living donor kidney exchange. Transplantation Proceedings, 37, pp. 589–591

Király, Z. (2012). Linear time local approximation algorithm for maximum stable marriage, in Proceedings of MATCH-UP '12: the 2nd International Workshop on Matching Under Preferences, pp. 99–110

Kobayashi, H. and Matsui, T. (2010). Cheating strategies for the Gale-Shapley algorithm with complete preference lists, Algorithmica 58, 1, pp. 151–169

University of Glasgow

References

Manlove, D.F., Irving, R.W., Iwama, K., Miyazaki, S. and Morita, Y. (1999). Hard variants of stable marriage, Tech. Rep. TR-1999-43, University of Glasgow, School of Computing Science

Manlove, D.F., Irving, R.W., Iwama, K., Miyazaki, S. and Morita, Y. (2002). Hard variants of stable marriage, Theoretical Computer Science 276, 1-2, pp. 261–279

Manlove, D.F. and McBride, I. (2013). The Hospitals / Residents problem with Couples, Unpublished manuscript

Manlove, D.F. and O'Malley, G. (2012). Paired and Altruistic Kidney Donation in the UK. In Proceedings of SEA 2012: the 11th International Symposium on Experimental Algorithms, Lecture Notes in Computer Science, Vol. 7276 (Springer), pp. 271-282

McDermid, E. (2009). A 3/2 approximation algorithm for general stable marriage, in Proceedings of ICALP '09: the 36th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 5555 (Springer), pp. 689–700

McDermid, E.J. and Manlove, D.F. (2010). Keeping partners together: Algorithmic results for the hospitals / residents problem with couples, Journal of Combinatorial Optimization 19, 3, pp. 279–303

University of Glasgow

References

Micali, S. and Vazirani, V.V. (1980). An O(√|V |·|E|) algorithm for finding maximum matching in general graphs, in Proceedings of FOCS ’80: the 21st Annual IEEE Symposium on Foundations of Computer Science (IEEE Computer Society), pp. 17–27.

Ng, C. and Hirschberg, D.S. (1988). Complexity of the stable marriage and stable roommate problems in three dimensions, Tech. Rep. UCI-ICS 88-28, Department of Information and Computer Science, University of California, Irvine

Paluch, K. (2012). Faster and simpler approximation of stable matchings, in Proceedings of WAOA ’11: 9th Workshop on Approximation and Online Algorithms, Lecture Notes in Computer Science, Vol. 7164 (Springer), pp. 176–187

Perach, N., Polak, J. and Rothblum, U.G. (2008). A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the Technion, International Journal of Game Theory 36, 3-4, pp. 519–535

Pini, M.S., Rossi, F., Venable, K.B. and Walsh, T. (2011). Manipulation complexity and gender neutrality in stable marriage procedures, Autonomous Agents and Multi-Agent Systems 22, 1, pp. 183–199

Rees, M.A., Kopke, J.E., Pelletier, R.P. et al. (2009). A nonsimultaneous, extended, altruistic-donor chain, New England Journal of Medicine, 360, pp. 1096–1101

University of Glasgow

References

Ronn, E. (1990). NP-complete stable matching problems, Journal of Algorithms 11, pp. 285–304

Roth, A.E. (1982). The economics of matching: Stability and incentives, Mathematics of Operations Research 7, 4, pp. 617–628

Roth, A.E. (1982a). Incentive compatibility in a market with indivisible goods, Economics Letters 9, pp. 127–132

Roth, A.E. (1984). The evolution of the labor market for medical interns and residents: a case study in game theory, Journal of Political Economy 92, 6, pp. 991–1016

Roth, A.E. (1986). On the allocation of residents to rural hospitals: a general property of two-sided matching markets, Econometrica 54, pp. 425–427

Roth, A.E. and Postlewaite, A. (1977). Weak versus strong domination in a market with indivisible goods, Journal of Mathematical Economics 4, pp. 131–137

Roth, A.E., Sönmez, T. and Ünver M.U. (2004). Kidney exchange. Quarterly Journal of Economics, 119, 2, pp. 457–488

University of Glasgow

References

Roth, A.E., Sönmez, T. and Ünver., M.U. (2005). Pairwise kidney exchange. Journal of Economic Theory, 125, pp. 151–188

Roth, A.E., Sönmez, T. and Ünver., M.U. (2007). Efficient kidney exchange: Coincidence of wants in a market with compatibility-based preferences. American Economic Review, 97, 3, 828–851

Teo, C.-P., Sethuraman, J. and Tan, W.-P. (1999). Gale-Shapley stable marriage problem revisited: strategic issues and applications, Management Science 47, 9, pp. 1252–1267

Toulis, P. and Parkes, D. (2011). A random graph model of kidney exchanges: efficiency, individual rationality and incentives, in Proceedings of the 12th ACM conference on Electronic commerce (ACM), pp. 323–332

Yanagisawa, H. (2007). Approximation Algorithms for Stable Marriage Problems, Ph.D. thesis, Kyoto University, School of Informatics

Yuan, Y. (1996). Residence exchange wanted: a stable residence exchange problem, European Journal of Operational Research 90, pp. 536–546