Algorithmic Game Theory

COMP6207

Lecture 14: **Stable Roommates**

Baharak Rastegari
b.rastegari@soton.ac.uk
Electronics and Computer Science
University of Southampton

Learning Outcomes

- By the end of this session, you should be able to
  - ***Describe*** the stable roommate problem and its objective.
  - ***Identify*** blocking pairs in an instance of SR
  - ***Compute*** a stable matching using Irving's algorithm
  - ***Describe*** the extensions of SR and their corresponding results

Stable Roommates

- School starts, find your roommate (even number of students)
- Each of you have a strict preference list
- Same notion of stability as stable marriage
- Not a bipartite graph anymore
- Stable matching (a.k.a. stable marriage) is special case

Stable Roommates problem (SR)

- **Participants**
  - **2n** students or agents
- **Preferences**
  - Each agent has strict preferences over all other (**2n-1**) agents
- **Matching**
  - A set of **n** disjoint pairs of agents
- **Stable Matching**
  - A matching **M** with no blocking pair
  - That is, no **{p,q}**$\notin$ **M** such that **p** prefers **q** to her partner in **M**, and **q** prefers **p** to her partner in **M**

Example

Example SR instance I_1:

A: C B D
B: D C A
C: B A D
D: A C B

The matching is not stable as **{A,C}** blocks.

Stable matching in I_1 :

A: C B D
B: D C A
C: B A D
D: A C B

A stable matching not always exists

- Consider the following instance:
  - A: B > C > D
  - B: C > A > D
  - C: A > B > D
  - D: A > B > C
- No stable matching in this case, why?

Irving's Algorithm

Knuth (1976): is there an efficient algorithm for deciding whether there exists a stable matching, given an instance of **SR**?

Irving (1985): ``**An efficient algorithm for the Stable Roommates problem**'', *Journal of Algorithms.*

- Given an instance of SR, decides whether a stable matching exists
- If so, finds one.

Irving's algorithm runs in $\mathbf{O(\mathit{n}^2)}$ time and works in two phases

- Phase 1: similar to GS algorithm for the Stable Matching problem
- Phase 2: elimination of "rotations"

Irving's Algorithm for SR

Semiengagement!

- Engagement is not a symmetric relation (unlike in SM)
  - If **x** is engaged to **y**, it is not necessarily the case that **y** is engaged to **x**

  - So we use the term semiengage and say that ``**x** is ***semiengaged*** to **y**''

  - **y** could be free, semiengaged to another agent **z**, or semiengaged to **x**

Free or not free

- All agents start out free
- A semiengaged agent **x** becomes free when her semi-fiancé rejects **x**
- **x** becomes semiengaged when she proposes to someone
  - The algorithm is designed so that when **x** proposes to **y** it must be the case that **y** is either free or finds **x** preferable to the current proposal she is holding

Irving's algorithm: Phase 1

Similar to Gale-Shapley

```
Phase 1 (agents, preferences)
    Assign all agents to be free; //initial state
    While (some free agent x has a nonempty list)
        y = first agent on x's list
        // next: x proposes to y
        If (some person z is semiengaged to y)
            assign z to be free; // y rejects z
        Assign x to be semiengaged to y;
        For (each successor w of x on y's list)
            delete w from y's list;
            delete y from w's list;
```

Example

- A: C > D > B > F > E
- B: F > E > D > A > C
- C: B > D > E > A > F
- D: E > B > C > F > A
- E: C > A > B > D > F
- F: E > A > C > D > B

Phase 1:

- Every agent proposes to others in order.
- Recipients hold their best proposers.

From now on we use the following convention:

$\underline{X}$ means **X** is semiengaged

$\overline{Y}$ means there is a **X** that is semiengaged to **Y**

Example: Phase 1

- $A: \cancel{\overline{C}} > \overline{D} > B > \underline{F} > \cancel{E}$
- $B: \overline{F} > E > D > A > \underline{C}$
- $C: \overline{B} > D > \underline{E} > \cancel{\underline{A}} > \cancel{F}$
- $D: \overline{E} > B > C > F > \underline{A}$
- $E: \overline{C} > \cancel{A} > B > \underline{D} > \cancel{F}$
- $F: \cancel{E} > \overline{A} > \cancel{C} > D > \underline{B}$

Phase 1:

- Every agent proposes to others in order.
- Recipients hold their best proposers.

From now on we use the following convention:

$\underline{\mathbf{X}}$ means **X** is semiengaged

$\overline{\mathbf{Y}}$ means there is a **X** that is semiengaged to **Y**

Example: Phase 1

- A: $\overline{D} > B > \underline{F}$
- B: $\overline{F} > E > D > A > \underline{C}$
- C: $\overline{B} > D > \underline{E}$
- D: $\overline{E} > B > C > F > \underline{A}$
- E: $\overline{C} > B > \underline{D}$
- F: $\overline{A} > D > \underline{B}$

After Phase 1, there are two possibilities:

1) Exist an agent who got rejected by everyone else.
$\Rightarrow$ No stable matching

2) Every agent holds a proposal. In other words, each row should have one top bar and one bottom bar (could coincide)
   i. If every row has only one remaining entry (top bar and bottom bar coincide) then we have a unique stable matching.
   ii. $\Rightarrow$ Phase 2

Phase 2: intuition

- The reduced preference lists (with deleted agents removed) at the end of phase 1 is referred to as **phase-1 table**

A	D B F
B	F E D A C
C	B D E
D	E B C F A
E	C B D
F	A D B

Phase 2: summary

- The reduced preference lists (with deleted agents removed) at the end of phase 1 is referred to as **phase-1 table**

- In phase 2, the preference table is further reduced until
  - All lists contain just one entry, in which case it constitutes a stable matching, or
  - Until some list becomes empty, in which case no stable matching exists.

- **Reduction** of preference table takes place by eliminating so called **``exposed rotations''**

Phase 2: intuition

- High-level intuition is that ``the existence of a stable matching is preserved by removing exposed rotations''.

That is:

- If the original preference table admits a stable matching, so does all subsequent reduced tables.
- If a reduced table doesn't admit a stable matching, neither does the original preference table.

Exposed Rotation

- A pair of sequences $\rho = ((p_0, \ldots, p_{r-1}), (q_0,\ldots,q_{r-1}))$ constitute an ***exposed rotation*** if
  - q_i is the second entry on p_i's list, and
  - p_{i+1} is the last entry on q_i's list

i+1 is taken modulo r

- **Example**

A	D	B	F		
B	F	E	D	A	**C**
C	B	D	E		
D	E	B	C	F	**A**
E	C	B	D		
F	A	D	B		

p:	A	C	A	Last entry
q:	B	D		Second entry

Eliminating an exposed rotation

- A pair of sequences $\rho = ((p_0, \ldots, p_{r-1}), (q_0, \ldots, q_{r-1}))$ constitute an *exposed rotation* if
  - q_i is the second entry on p_i's list, and
  - p_{i+1} is the last entry on q_i's list

i+1 is taken modulo r

- Exposed rotation $\rho = ((p_0, \ldots, p_{r-1}), (q_0, \ldots, q_{r-1}))$ is **eliminated** by deleting (q_i, p_{i+1}) from each other's lists, for all i
  - delete q_i from p_{i+1}'s list
  - delete p_{i+1} from q_i 's list

Example: elimination

- Exposed rotation $\rho = ((p_0, \ldots, p_{r-1}), (q_0, \ldots, q_{r-1}))$ is **eliminated** by deleting (q_i, p_{i+1}) from each other's lists, for all i
  - delete q_i from p_{i+1}'s list
  - delete p_{i+1} from q_i 's list

A	~~D~~	B	F		
B	F	E	D	A	~~C~~
C	~~B~~	D	E		
D	E	B	C	F	~~A~~
E	C	B	D		
F	A	D	B		

p:	A	C	A	Last entry
q:	B	D		Second entry

Irving's algorithm: Phase 2

Reducing preference table

```
Phase 2 (phase-1 Table)
      T= phase-1 table; //initial state
      While ((some list in T has more than one entry)
and (no list in T is empty))
            find a rotation ρ exposed in T
            T = T / ρ // eliminate ρ
       If (some list in T is empty)
            return instance unsolvable;
       Else
            return T, which is a stable matching;
```

Example: Phase 2

- A: D > B > F
- B: F > E > D > A > C
- C: B > D > E
- D: E > B > C > F > A
- E: C > B > D
- F: A > D > B

First round:

Least: A C A

2^{nd}: B D

D will reject A! and B will reject C!

Example: Phase 2

- A: B > F
- B: F > E > D > A
- C: D > E
- D: E > B > C > F
- E: C > B > D
- F: A > D > B

First round:

Least: A C A

2nd: B D

D will reject A! and B will reject C!

Second round:

Least: A B D A

2nd: F E B

Example: Phase 2

- A: $\underline{F}$
- B: E > D
- C: D > $\underline{E}$
- D: B > C > F
- E: $\overline{C}$ > B
- F: $\overline{A}$ > D

First round:

Least: A C A

2nd: B D

D will reject A! and B will reject C!

Second round:

Least: A B D A

2nd: F E B

Example: Phase 2

- A: F
- B: E > D
- C: D > E
- D: B > C > F
- E: C > B
- F: A > D

First round:

Least:	A	C	A
2^{nd}:	B	D	

Second round:

Least:	A	B	D	A
2^{nd}:	F	E	B	

Third round:

Least:	B	F	F
2^{nd}:	D	D	

Cycle; eliminate (D, F) pair

Example: Phase 2

- A: F
- B: E > D
- C: D > E
- D: B > C
- E: C > B
- F: A

First round:
Least: A C A
2^{nd}: B D

Second round:
Least: A B D A
2^{nd}: F E B

Third round:
Least: B F F
2^{nd}: D D
Cycle; eliminate (D, F) pair

Example: Phase 2

- A: F
- B: E > D
- C: D > E
- D: B > C
- E: C > B
- F: A

First round:

Least:	A	C	A
2^{nd}:	B	D	

Second round:

Least:	A	B	D	A
2^{nd}:	F	E	B	

Third round:

Least:	B	F	F
2^{nd}:	D	D	

Cycle; eliminate (D, F) pair

Fourth round:

Least:	B	C	B
2^{nd}:	D	E	

Example: Phase 2

- A: F
- B: D
- C: E
- D: B
- E: C
- F: A

First round:

Least: A C A

2^{nd}: B D

Second round:

Least: A B D A

2^{nd}: F E B

Third round:

Least: B F F

2^{nd}: D D

Cycle; eliminate (D, F) pair

Fourth round:

Least: B C B

2^{nd}: D E

Quiz

- A: B > D > F > C > E
- B: D > E > F > A > C
- C: D > E > F > A > B
- D: F > C > A > E > B
- E: F > C > D > B > A
- F: A > B > D > C > E

Question

- If there is a stable matching, do all executions of Irving's algorithm return the same matching?

  - Phase 1 always returns the same preference table (the same phase-1 table)
  - Depending on the order of elimination in phase 2, we may arrive at different stable matchings.
  - So the answer is, no.

Question

- Is Irving's algorithm DS truthful?

- If not, how to find a best strategy to be matched with a roommate as good as possible?

Extensions of SR

1. Odd number of agents

- We might have odd number of agents, say **2n+1**
- In any matching one agent is unmatched

- Stability defined as before
- Recall that an agent prefers being matched than to remain unmatched

Extending Irving's algorithm

- If after **phase 1** all preference lists are nonempty then there exists no stable matching
  - Coursework 1 Exercise 4: prove the previous statement.

- If at the end of **phase 1** exactly one agent's, say **x**, preference list becomes empty
  - **x** cannot be matched in any stable matching
  - If there exists a stable matching, no other person is unmatched
  - To check whether or not there exists a stable matching, we run **phase 2** (without agent **x**)

After the execution of **phase 1**, at most one agent's preference list can be empty.
Proof: left as an exercise.

2. Unacceptable partners

- As in the marriage market, some agents might find some other agents unacceptable. In this case we get Stable Roommates with Incomplete lists (**SRI**)
- Matching is a pairing of agents
    - One or more agents may remain unmatched
- Stability defined as before
- Recall that an agent prefers being matched to an acceptable partner than to remain unmatched

Extending Irving’s algorithm

- After **phase 1** one or more preference lists may become empty.
  - Any agent with empty list cannot be matched in any stable matching.
  - Any person with nonempty list must be matched in every stable matching
  - To check whether or not there exists a stable matching, we run **phase 2** (without those agents whose lists are empty at the end of phase 1)

Stable Pairs

- Any pair (X, Y) removed during **phase 1** cannot be a stable pair
  - i.e. X and Y are not matched in any of the stable matchings

- For each agent, the first entry in the **phase-1 table** is their best achievable roommate (if there is any achievable roommate)
- For each agent, the last entry in the **phase-1 table** is their worst achievable roommate (if there is any achievable roommate)

3. Ties

- As in the marriage market, some agents might be indifferent between several other agents. In this case we get Stable Roommates with Ties (**SRT**)

- As in SMT, different types of blocking pairs can be defined for SRT. Here we will focus on the straightforward extension of the stability notion we already know
  - A matching is stable if there is no pair such that each of whom would strictly prefer to match with each other rather than their assigned partner.

SRT

- Can we decide in polynomial time whether an instance of SRT admits a stable matching? And if so, fine one?

Theorem (Ronn, 1990)

Deciding whether a stable matching exists, given an instance of SRT, is NP-complete.

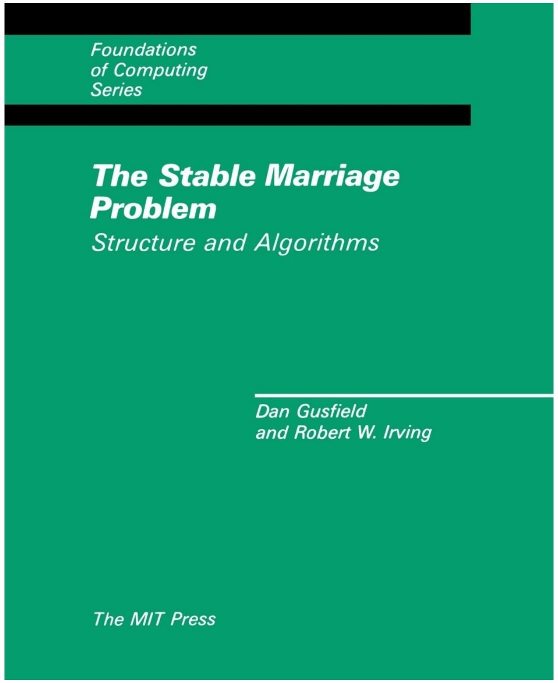
- Even if each preference list is either strictly ordered or contains a tie of length 2 at the head.

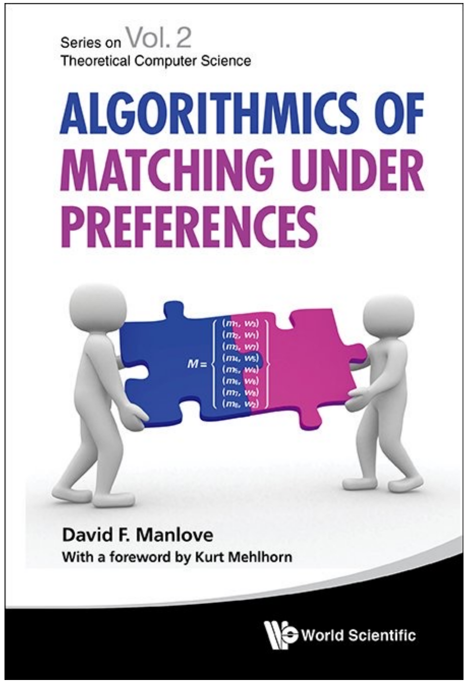
Acknowledgement

Some of the slides in this lecture were based on the slides by **David Manlove** and **Jie Zhang**.

Books

- The Stable Marriage Problem - Structure and Algorithms by Dan Gusfield and Robert W. Irving





- Algorithmics of Matching under Preferences by David F. Manlove.

Optional additional reading

- Chapter 4 of The Stable Marriage Problem - Structure and Algorithms by Dan Gusfield and Robert W. Irving