CSE 421: Introduction to Algorithms

Stable Matching

Shayan Oveis Gharan

Summary

Stable matching problem: Given n men and n women, and their preferences, find a stable matching if one exists.

- Gale-Shapley algorithm: Guarantees to find a stable matching for any problem instance.
- Q: How to implement GS algorithm efficiently?
- Q: If there are multiple stable matchings, which one does GS find?
- Q: How many stable matchings are there?

Propose-And-Reject Algorithm [Gale-Shapley'62]

```
Initialize each person to be free.
while (some man is free and hasn't proposed to every woman) {
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
}
```

Implementation of GS Algorithm

Problem size

$N=2n^2$ words

- $2n$ people each with a preference list of length n

Brute force algorithm

Try all $n!$ possible matchings

Do any of them work?

Gale-Shapley Algorithm

n^2 iterations, each costing constant time as follows:

Efficient Implementation

We describe $O(n^2)$ time implementation.

Representing men and women:

Assume men are named 1, …, n.
Assume women are named n+1, …, 2n.

Engagements.

Maintain a list of free men, e.g., in a queue.
Maintain two arrays **wife[m]**, and **husband[w]**.

- set entry to **0** if unmatched
- if **m** matched to **w** then **wife[m]=w** and **husband[w]=m**

Men proposing:

For each man, maintain a list of women, ordered by preference.
Maintain an array **count[m]** that counts the number of proposals made by man **m**.

Efficient Implementation

Women rejecting/accepting.

Does woman **w** prefer man **m** to man **m'**?

For each woman, create inverse of preference list of men.

Constant time access for each query after O(n) preprocessing per woman.

O(n^2) total reprocessing cost.

w_i	1st	2nd	3rd	4th	5th	6th	7th	8th
Pref	8	3	7	1	4	5	6	2

w_i	1	2	3	4	5	6	7	8
Inverse	4th	8th	2nd	5th	6th	7th	3rd	1st

```
for i = 1 to n
   for j = 1 to n
      inverse[i][pref[i][j]] = j
```

w_i prefers man 3 to 6

since `inverse[i][3]`=2 < 7=`inverse[i][6]`

Summary

- Stable matching problem: Given **n** men and **n** women, and their preferences, find a stable matching if one exists.
- Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.
- GS algorithm finds a stable matching in $\mathbf{O}(\mathbf{n}^2)$ time.
- Q: If there are multiple stable matchings, which one does GS find?
- Q: How many stable matchings are there?

Understanding the Solution

Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

An instance with two stable matchings:

- (m_1, w_1), (m_2, w_2).
- (m_1, w_2), (m_2, w_1).

	1st	2nd
m_1	w_1	w_2
m_2	w_2	w_1

	1st	2nd
w_1	m_2	m_1
w_2	m_1	m_2

Man Optimal Assignments

Definition: Man m is a valid partner of woman w if there exists some stable matching in which they are matched.

Man-optimal matching: Each man receives the best valid partner (according to his preferences).

- Not that each man receives his most favorite woman.

Example

Here

Valid-partner(m_1) $= \{w_1, w_2\}$

Valid-partner(m_2) $= \{w_1, w_2\}$

Valid-partner(m_3) $= \{w_3\}$.

Man-optimal matching $\{m_1, w_1\}, \{m_2, w_2\}, \{m_3, w_3\}$

	1st (favorite)	2nd	3rd (least favorite)
m_1	w_1	w_2	w_3
m_2	w_2	w_1	w_3
m_3	w_1	w_2	w_3

	1st (favorite)	2nd	3rd (least favorite)
w_1	m_2	m_1	m_3
w_2	m_1	m_2	m_3
w_3	m_1	m_2	m_3

Man Optimal Assignments

Definition: Man m is a valid partner of woman w if there exists some stable matching in which they are matched.

Man-optimal matching: Each man receives the best valid partner (according to his preferences).

- Not that each man receives his most favorite woman.

Claim: All executions of GS yield a man-optimal matching, which is a stable matching!

- So, output of GS is unique!!
- No reason a priori to believe that man-optimal matching is perfect, let alone stable.

Man Optimality

S
(m, w)
(m', w')
$\cdots$

Claim: GS matching **S*** is man-optimal.

Proof: (by contradiction)

Suppose some man is paired with someone other than his best partner. Men propose in decreasing order of preference $\Rightarrow$ some man is rejected by a valid partner.

Let m be the man who is the first such rejection, and let w be the women who is first valid partner that rejects him.

Let **S** be a stable matching where m and w are matched.

In building **S***, when m is rejected, w forms (or reaffirms) engagement with a man, say m' whom she prefers to m.

Let w' be m' partner in **S**.

In building **S***, m' is not rejected by any valid partner at the point when m is rejected by w. Thus, m' prefers w to w'.

since this is the first rejection by a valid partner

But w prefers m' to m.

Thus (m', w) is unstable in **S**. ■

Man Optimality Summary

Man-optimality: In version of GS where men propose, each man receives the best valid partner.

> w is a valid partner of m if there exist some stable matching where m and w are paired

Q: Does man-optimality come at the expense of the women?

Woman Pessimality

Woman-pessimal assignment: Each woman receives the worst valid partner.

Claim. GS finds woman-pessimal stable matching **S***.

Proof.

Suppose (m, w) matched in **S***, but m is not worst valid partner for w.

There exists stable matching **S** in which w is paired with a man, say m', whom she likes less than m.

- Let w' be m partner in **S**.
- m prefers w to w'. ← man-optimality of **S***
- Thus, (m, w) is an unstable in **S**.

■

Summary

- Stable matching problem: Given **n** men and **n** women, and their preferences, find a stable matching if one exists.
- Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.
- GS algorithm finds a stable matching in $\mathbf{O}(\mathbf{n}^2)$ time. ✓
- GS algorithm finds man-optimal woman pessimal matching ✓
- Q: How many stable matching are there?

How many stable Matchings?

We already show every instance has at least 1 stable matchings.

There are instances with about b^n stable matchings for $b > 2$

[Karlin-O-Weber'17]: Every instance has at most c^n stable matchings for some $c > 2$

[Research-Question]:
Is there an "efficient" algorithm that chooses a uniformly random stable matching of a given instance.

Extensions: Matching Residents to Hospitals

Men ≈ hospitals, Women ≈ med school residents.

- Variant 1: Some participants declare others as unacceptable.
- Variant 2: Unequal number of men and women.
- Variant 3: Limited polygamy.

e.g. A resident not interested in Cleveland

e.g. A hospital wants to hire **3** residents

Def: Matching **S** is unstable if there is hospital **h** and resident **r** s.t.

- **h** and **r** are acceptable to each other; and
- either **r** is unmatched, or **r** prefers **h** to her assigned hospital; and
- either **h** does not have all its places filled, or **h** prefers **r** to at least one of its assigned residents.

Lessons Learned

- Powerful ideas learned in course.
  - Isolate underlying structure of problem.
  - Create useful and efficient algorithms.

- Potentially deep social ramifications. [legal disclaimer]
  - Historically, men propose to women. Why not vice versa?
  - Men: propose early and often.
  - Men: be more honest.
  - Women: ask out the guys.
  - Theory can be socially enriching and fun!

“The Match”: Doctors and Medical Residences

- Each medical school graduate submits a ranked list of hospital where he wants to do a residency

- Each hospital submits a ranked list of newly minted doctors
- A computer runs stable matching algorithm (extended to handle polygamy)
- Until recently, it was hospital-optimal.

History

1900

- Idea of hospital having residents (then called “interns”)

1900-1940s

- Intense competition among hospitals
  - Each hospital makes offers independently
  - Process degenerates into a race; hospitals advancing date at which they finalize binding contracts

1944

- Medical schools stop releasing info about students before a fixed date

1945-1949

- Hospitals started putting time limits on offers
  - Time limits down to 12 hours; lots of unhappy people

"The Match"

1950

- NICI run a centralized algorithm for a trial run
- The pairing was not stable, Oops!!

1952

- The algorithm was modified and adopted. It was called the Match.
- The first matching produced in April 1952

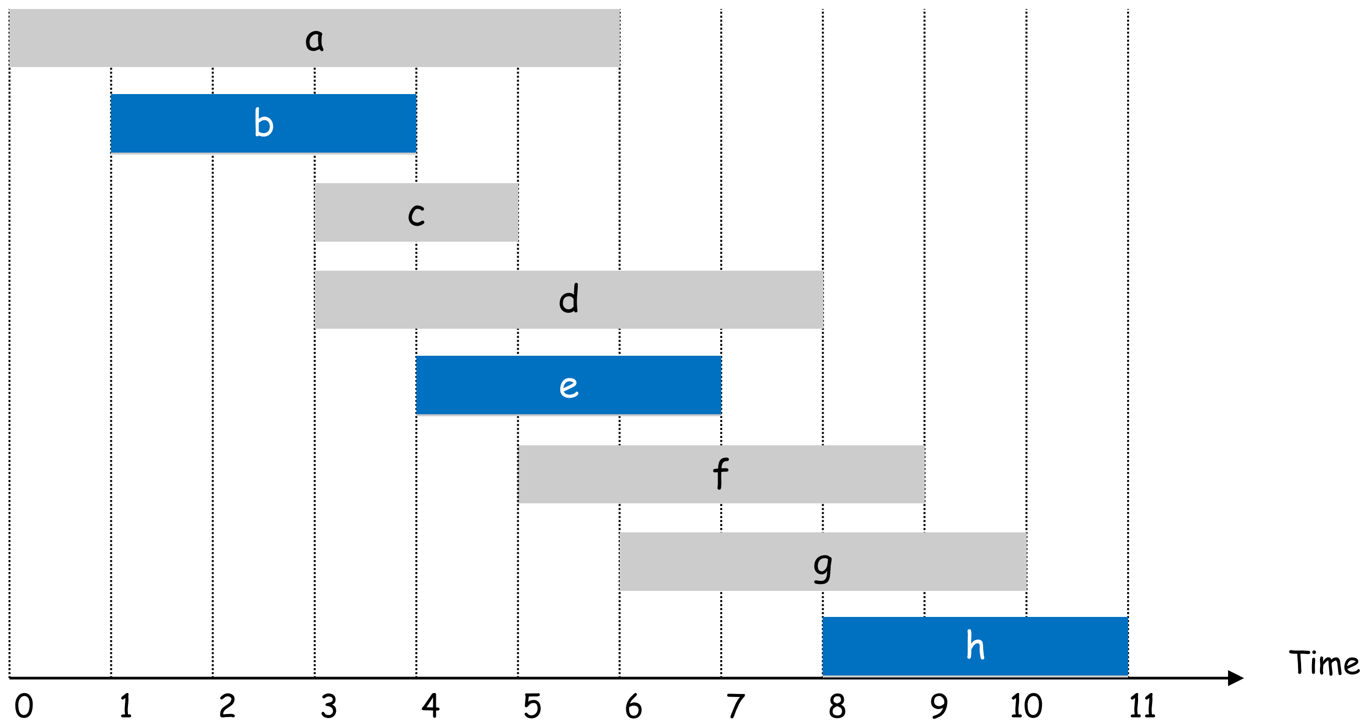
Five Representative Problems

1. Interval Scheduling
2. Weighted Interval Scheduling
3. Bipartite Matching
4. Independent Set Problem
5. Competitive Facility Location

Interval Scheduling

Input: Given a set of jobs with start/finish times

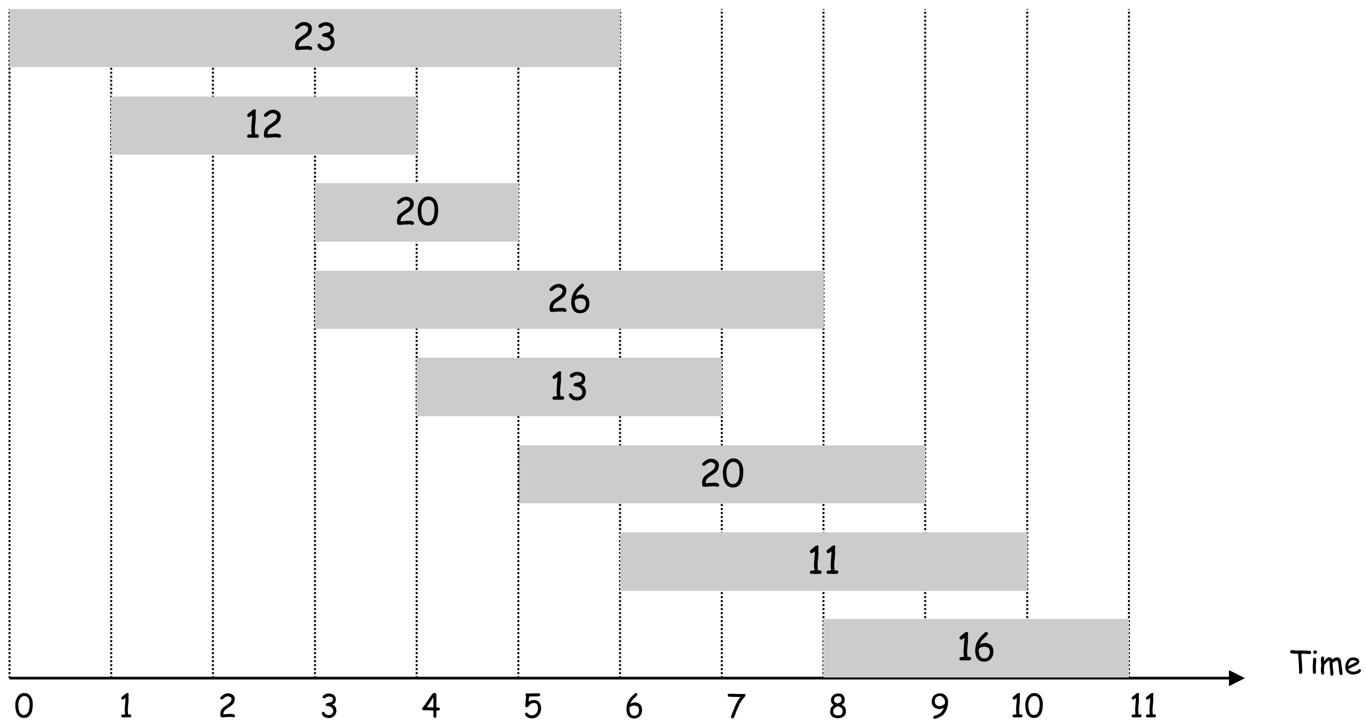
Goal: Find the maximum cardinality subset of jobs that can be run on a single machine.



Interval Scheduling

Input: Given a set of jobs with start/finish times

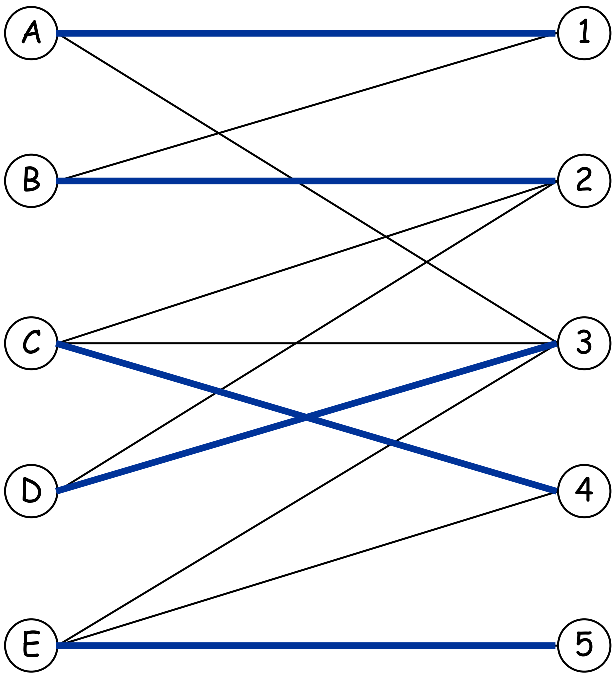
Goal: Find the maximum weight subset of jobs that can be run on a single machine.



Bipartite Matching

Input: Given a bipartite graph

Goal: Find the maximum cardinality matching

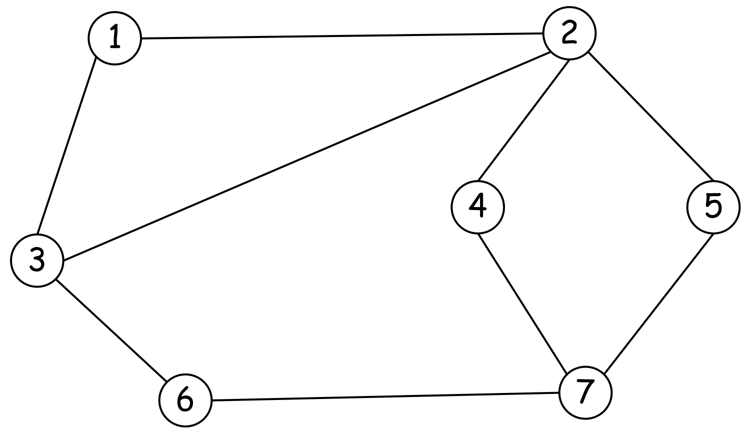


Independent Set

Input: A graph

Goal: Find the maximum independent set

Subset of nodes that no two joined by an edge

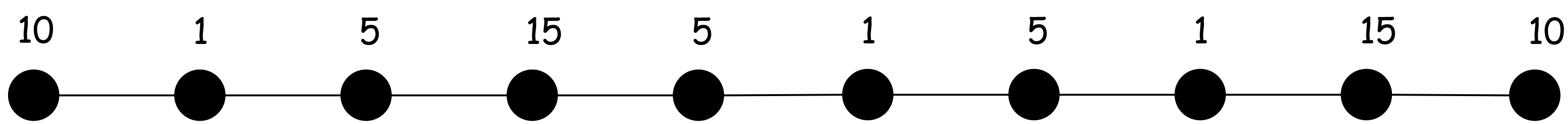


Competitive Facility Location

Input: Graph with weight on each node

Game: Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors have been selected.

Goal. Does player 2 have a strategy which guarantees a total value of V no matter what player 1 does?



Second player can guarantee 20, but not 25.

Five Representative Problems

Variation of a theme: Independent set Problem

1. Interval Scheduling
   $n \log n$ greedy algorithm

2. Weighted Interval Scheduling
   $n \log n$ dynamic programming algorithm

3. Bipartite Matching
   n^k maximum flow based algorithm

4. Independent Set Problem: NP-complete

5. Competitive Facility Location: PSPACE-complete