1.1 A First Problem: Stable Matching

Goal. Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process.

**Unstable pair**: applicant \( x \) and hospital \( y \) are unstable if:
- \( x \) prefers \( y \) to its assigned hospital.
- \( y \) prefers \( x \) to one of its admitted students.

**Stable assignment**: Assignment with no unstable pairs.
- Natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital deal from being made.

Matching Residents to Hospitals

**Goal.** Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process.

**Unstable pair**: applicant \( x \) and hospital \( y \) are unstable if:
- \( x \) prefers \( y \) to its assigned hospital.
- \( y \) prefers \( x \) to one of its admitted students.

**Stable assignment**: Assignment with no unstable pairs.
- Natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital deal from being made.

**Stable Matching Problem**

**Goal.** Given \( n \) men and \( n \) women, find a "suitable" matching.
- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

**Natural and desirable condition.** Individual self-interest will prevent any applicant/hospital deal from being made.

**Stable Matching Problem**

**Goal.** Given \( n \) men and \( n \) women, find a "suitable" matching.
- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

**Stability**: no incentive for some pair of participants to undermine assignment by joint action.
- In matching \( M \), an unmatched pair \( m-w \) is unstable if man \( m \) and woman \( w \) prefer each other to current partners.
- Unstable pair \( m-w \) could each improve by changing.

**Stable matching**: perfect matching with no unstable pairs.

**Stable matching problem.** Given the preference lists of \( n \) men and \( n \) women, find a stable matching if one exists.
Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?
A. No. Bertha and Xavier will hook up.

Stable Matching Problem

Q. Is assignment X-A, Y-B, Z-C stable?
A. Yes.

Stable Roommate Problem

Q. Do stable matchings always exist?
A. Not obvious a priori.

Stable roommate problem.

2n people; each person ranks others from 1 to 2n-1.

Assign roommate pairs so that no unstable pairs.

Observation. Stable matchings do not always exist for stable roommate problem.


Intuitive method that guarantees to find a stable matching.

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
}

Proof of Correctness: Termination

Observation 1. Men propose to women in decreasing order of preference.
Observation 2. Once a woman is matched, she never becomes unmatched; she only "trades up."

Claim. Algorithm terminates after at most n^2 iterations of while loop.

Pf. Each time through the while loop a man proposes to a new woman.

There are only n possible proposals.

n(n-1) + 1 proposals required

Proof of Correctness: Perfection

Claim. All men and women get matched.

Pf. (by contradiction)

Suppose, for sake of contradiction, that Zeus is not matched upon termination of algorithm.

Then some woman, say Amy, is not matched upon termination.

By Observation 2, Amy was never proposed to.

But, Zeus proposes to everyone, since he ends up unmatched.

n^2 + 1 proposals required
Proof of Correctness: Stability

Claim. No unstable pairs.

Pf. (by contradiction)

Suppose A-Z is an unstable pair: each prefers each other to partner in Gale-Shapley matching S*.

Case 1: Z never proposed to A.
- Z prefers his GS partner to A.
- A-Z is stable.

Case 2: Z proposed to A.
- A rejected Z (right away or later)
- A prefers her GS partner to Z.
- A-Z is stable.

In either case A-Z is stable, a contradiction.

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?

Efficient Implementation

Efficient implementation. We describe O(n²) time implementation.

Representing men and women.
- Assume men are named 1, ..., n.
- Assume women are named 1', ..., n'.

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.

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.

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.
- A-X, B-Y, C-Z.
- A-Y, B-X, C-Z.

Q. 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.

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?

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

Man-optimal assignment. Each man receives best valid partner.

Clm. All executions of GS yield man-optimal assignment, which is a stable matching!

- No reason a priori to believe that man-optimal assignment is perfect, let alone stable.
- Simultaneously best for each and every man.
Man Optimality

Claim. GS matching $S^*$ is man-optimal.

Pf. (by contradiction)

Suppose some man is paired with someone other than best partner.

Men propose in decreasing order of preference — some man is rejected by valid partner.

Let $Y$ be first such man, and let $A$ be first valid woman that rejects him.

Let $S$ be a stable matching where $A$ and $Y$ are matched.

When $Y$ is rejected, $A$ forms (or reaffirms) engagement with a man, say $Z$, whom she prefers to $Y$.

Let $B$ be $Z$'s partner in $S$.

$Z$ not rejected by any valid partner at the point when $Y$ is rejected by $A$. Thus, $Z$ prefers $A$ to $B$.

But $A$ prefers $Z$ to $Y$.

Thus $A-Z$ is unstable in $S$.

Stable Matching Summary

Stable matching problem. Given preference profiles of $n$ men and $n$ women, find a stable matching.

Gale-Shapley algorithm. Finds a stable matching in $O(n^2)$ time.

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

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

Woman Pessimality

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

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

Pf. Suppose $A-Z$ matched in $S^*$, but $Z$ is not worst valid partner for $A$.

There exists stable matching $S$ in which $A$ is paired with a man, say $Y$, whom she likes less than $Z$.

Let $B$ be $Z$'s partner in $S$.

$Z$ prefers $A$ to $B$.

Thus, $A-Z$ is unstable in $S$.

Extensions: Matching Residents to Hospitals

Ex: 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.

Def. Matching $S$ unstable if there is a hospital $h$ and resident $r$ such that:

- $h$ 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.

Application: Matching Residents to Hospitals

NRMP. (National Resident Matching Program)

- Original use just after WWII.
- Idea of March, 23,000+ residents.

Rural hospital dilemma.

Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.

Rural hospitals were under-subscribed in NRMP matching.

Rural Hospital Theorem. Rural hospitals get exactly same residents in every stable matching!

Lessons Learned

Powerful ideas learned in course.

- Isolate underlying structure of problem.
- Create useful and efficient algorithms.

Potentially deep social ramifications. (legal disclaimer)
1.2 Five Representative Problems

**Interval Scheduling**
- Input: Set of jobs with start times and finish times.
- Goal: Find maximum cardinality subset of mutually compatible jobs.
- Jobs don't overlap.

**Weighted Interval Scheduling**
- Input: Set of jobs with start times, finish times, and weights.
- Goal: Find maximum weight subset of mutually compatible jobs.

**Bipartite Matching**
- Input: Bipartite graph.
- Goal: Find maximum cardinality matching.

**Independent Set**
- Input: Graph.
- Goal: Find maximum cardinality independent set.

**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: Select a maximum weight subset of nodes.

Second player can guarantee 20, but not 25.
Five Representative Problems
Variations on a theme: independent set.
Internal scheduling: $n \log n$ greedy algorithm.
Weighted internal scheduling: $n \log n$ dynamic programming algorithm.
Bipartite matching: $n^2$ max-flow based algorithm.
Independent set: NP-complete.
Competitive facility location: PSPACE-complete.

Extra Slides
Stable Matching Problem
Goal: Given $n$ men and $n$ women, find a “suitable” matching.
- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

ZEUS
BERtha
ALY
Diana
ERIKA
CLARe

YANCEY
ALY
CLARe
Diana
ERIKA
BERtha

XAVIER
BERtha
CLARe
Diane
ERIKA
Diana

VICTOR
BERtha
Diane
ALY
ERIKA
CLARe

WYATT
Diane
ALY
BERtha
ERIKA

Women’s Preference List

Men’s Preference List

Understanding the Solution
Claim: The man-optimal stable matching is weakly Pareto optimal.

Pf.
- Let $A$ be last woman in some execution of GS algorithm to receive a proposal.
- No man is rejected by $A$ since algorithm terminates when last woman receives first proposal.
- No man matched to $A$ will be strictly better off than in man-optimal stable matching.

Deceit: Machiavelli Meets Gale-Shapley
Q. Can there be an incentive to misrepresent your preference profile?
- Assume you know men’s propose-and-reject algorithm will be run.
- Assume that you know the preference profiles of all other participants.

Fact: No, for any man yes, for some women. No mechanism can guarantee a stable matching and be cheatproof.
Lessons Learned

Powerful ideas learned in course.
- Isolate underlying structure of problem.
- Create useful and efficient algorithms.
- 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.
- CS majors get the best partners.

4.5 Minimum Spanning Tree

Minimum spanning tree. Given a connected graph $G = (V, E)$ with real-valued edge weights $c_e$, an MST is a subset of the edges $T \subseteq E$ such that $T$ is a spanning tree whose sum of edge weights is minimized.

Applications

MST is a fundamental problem with diverse applications.
- Network design.
  - Telephone, electrical, hydraulic, TV cable, computer, road
- Approximation algorithms for NP-hard problems.
  - Traveling salesman problem, shortest tree
- Indirect applications.
  - Max bottleneck path
  - LDPC code for error correction
  - Image registration with Rényi entropy
  - Learning robust features for real-time face verification
  - Reducing data storage in sequencing amino acids in a protein
  - Modeling chaotic particle interactions in turbulent fluid flows
  - Autoconfig protocol for Ethernet bridging to avoid cycles in a network
- Cluster analysis.

Greedy Algorithms

Kruskal’s algorithm. Start with $T = \emptyset$. Consider edges in ascending order of cost. Insert edge $e$ in $T$ unless doing so would create a cycle.

Reverse-Delete algorithm. Start with $T = E$. Consider edges in descending order of cost. Delete edge $e$ from $T$ unless doing so would disconnect $T$.

Prim’s algorithm. Start with some root node $s$ and greedily grow a tree $T$ from $s$ outward. At each step, add the cheapest edge $e$ to $T$ that has exactly one endpoint in $T$.

Remark. All three algorithms produce an MST.

Greedy Algorithms

Simplifying assumption. All edge costs $c_e$ are distinct.

Cut property. Let $S$ be any subset of nodes, and let $e$ be the min cost edge with exactly one endpoint in $S$. Then the MST contains $e$.

Cycle property. Let $C$ be any cycle, and let $f$ be the max cost edge belonging to $C$. Then the MST does not contain $f$.
Cycles and Cuts

Cycle. A set of edges of the form a-b, b-c, c-d, ..., y-z, z-a.

Cutset. A cut is a subset of nodes S. The corresponding cutset D is the subset of edges with exactly one endpoint in S.

**Example:**

Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1

Cut S = \{4, 5, 8\}

Cutset D = 5-6, 5-7, 3-4, 3-5, 7-8

Cycle-Cut Intersection

**Claim.** A cycle and a cutset intersect in an even number of edges.

**Proof.** (by picture)

Greedy Algorithms

Simplifying Assumption. All edge costs \(c_e\) are distinct.

**Cycle Property.** Let C be any cycle in \(G\), and let \(f\) be the max cost edge belonging to C. Then the MST \(T^*\) does not contain \(f\).

**Proof.** (exchange argument)

Suppose \(f\) belongs to \(T^*\), and let’s see what happens.

Deleting \(f\) from \(T^*\) creates a cut \(S\) in \(T^*\).

Edge \(f\) is both in the cycle \(C\) and in the cutset \(D\) corresponding to \(S\).

There exists another edge, say \(e\), that is in both \(C\) and \(D\).

\(T^* = T^* \setminus \{f\} \cup \{e\}\) is also a spanning tree.

Since \(c_e < c_f\), cost\((T^*) < cost(T^*)\).

This is a contradiction.

Prim’s Algorithm: Proof of Correctness

**Prim’s Algorithm.** [Zarlič 1930, Dijkstra 1957, Prim 1959]

1. Initialize \(S\) = any node.
2. Apply cut property to \(S\).
3. Add min cost edge in cutset corresponding to \(S\) to \(T\), and add one new explored node \(u\) to \(S\).

**Implementation.** Prim’s Algorithm

Implementation. Use a priority queue.

1. Maintain set of explored nodes \(S\).
2. For each unexplored node \(v\), maintain attachment cost \(a[v] = \text{cost of cheapest edge } v \text{ to a node in } S\).
3. \(O(n \log n)\) with a binary heap.
Kruskal’s Algorithm: Proof of Correctness

Kruskal’s algorithm. [Kruskal, 1956]

1. Consider edges in ascending order of weight.
2. Case 1: If adding e to T creates a cycle, discard e according to cycle property.
3. Case 2: Otherwise, insert e = (u, v) into T according to cut property where S = set of nodes in u’s connected component.

Implementation: Kruskal’s Algorithm

Implementation. Use the union-find data structure.

1. Build set T of edges in the MST.
2. Maintain set for each connected component.
3. O(m log n) for sorting and O(m + n) for union-find.

Lexicographic Tiebreaking

To remove the assumption that all edge costs are distinct: perturb all edge costs by tiny amounts to break any ties.

Implementation. Can handle arbitrarily small perturbations implicitly by breaking ties lexicographically, according to index.

boolean less(i, j) {
    if      (cost(ei) < cost(ej)) return true
    else if (cost(ei) > cost(ej)) return false
    else if (i < j)               return true   else                          return false
}

Algorithms

- Greedy
  - MST
- Divide and conquer
  - mergesort, heapsort
- Dynamic programming
  - Shortest paths, fibonacci numbers, knapsack, subset sum
- Max flow
- Intractable problems
  - Approximation
  - Randomization
  - Parameterized or sub-problem solutions