Lecture 1: Introduction - Some Representative Problems

subject	TDT4121
topics	Bipartite Matching, Gale-Shapley Algorithm, Independent Set, Interval Scheduling Problem, Stable Matching Problem
created	@August 26, 2022 8:17 AM

Chapter 1 - Introduction: Some Representative Problems

> We start with an introduction of several representative problems, which introduce you to the world of algorithms.

Goals 🏁

The student should be able to:

- Understand the *Stable Matching Problem* ✅
- Understand how the *Gale-Shapley algorithm* solves the stable matching problem ✅
- Understand some representative problems ✅

Stable Matching Problem

> Stable matching involves finding a stable match for two sets of elements with equal size, given a preference order for each element. This problem is usually presented as the stable marriage problem, where given N men and N women where each person has ranked all members of opposite sex in order of

> preference. The men and women have to get married so that there are no two people of opposite sex who would both rather have each other than their current partners. If there exists a matching where a pair prefer each other to their current partners, then the matching is not stable.

Perfect matching: everyone is matched monogamously.

- Each man gets exactly one woman.
- Each woman gets exactly one man.

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 eloping.

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.

Gale-Shapley Algorithm

The algorithm finds the solution to the stable matching problem. It takes polynomial time.

The **Gale–Shapley algorithm** involves a number of "rounds" (or "iterations"):

- In the first round, first *a*) each unengaged man proposes to the woman he prefers most, and then *b*) each woman replies "maybe" to her suitor she most prefers and "no" to all other suitors. She is then provisionally "engaged" to the suitor she most prefers so far, and that suitor is likewise provisionally engaged to her.
- In each subsequent round, first *a*) each unengaged man proposes to the most-preferred woman to whom he has not yet proposed (regardless of whether the woman is already engaged), and then *b*) each woman replies "maybe" if she is currently not engaged or if she prefers this man over her current provisional partner (in this case, she rejects her current provisional partner who becomes unengaged). The provisional nature of engagements preserves the right of an

already-engaged woman to "trade up" (and, in the process, to "jilt" her until-then partner).

- This process is repeated until everyone is engaged.

```
#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 man
}
```

Man optimality

> Claim: GS matching S* is man-optimal

Suppose some man is paired with someone other than best partner. Men propose in decreasing order of preferences ⇒ some man is rejected by valid partner

- Y is the first such man
- A is the first valid woman that rejects him
- S is 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

- B is Z’s partner in S

Z is 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.

Interval Scheduling

Interval scheduling is a class of problems in computer science, particularly in the area of algorithm design. The problems consider a set of tasks. Each task is represented by an *interval* describing the time in which it needs to be processed by some machine (or, equivalently, scheduled on some resource). For instance, task A

might run from 2:00 to 5:00, task B might run from 4:00 to 10:00 and task C might run from 9:00 to 11:00. A subset of intervals is *compatible* if no two intervals overlap on the machine/resource. For example, the subset {A,C} is compatible, as is the subset {B}; but neither {A,B} nor {B,C} are compatible subsets, because the corresponding intervals within each subset overlap.

Read more about the interval scheduling problem in Lecture 5: Greedy Algorithms

Weighted Interval Scheduling

When the intervals have weights, the problem is equivalent to finding a maximum-weight independent set in an interval graph. It can be solved in polynomial time.

Independent set

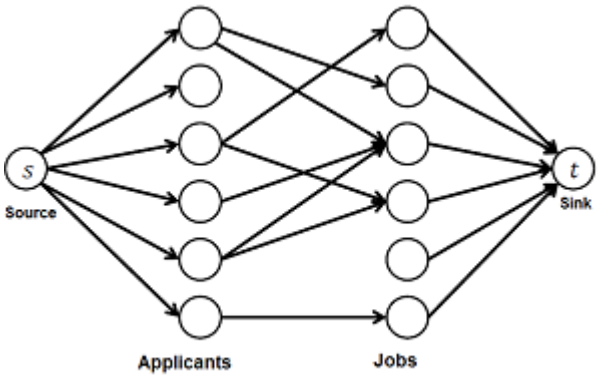
In graph theory, an **independent set** is a set of vertices in a graph, no two of which are adjacent. That is, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two. Equivalently, each edge in the graph has at most one endpoint in S. A set is independent if and only if it is a clique in the graph's complement. The size of an independent set is the number of vertices it contains. Independent sets have also been called "internally stable sets", of which "stable set" is a shortening

Bipartite Matching

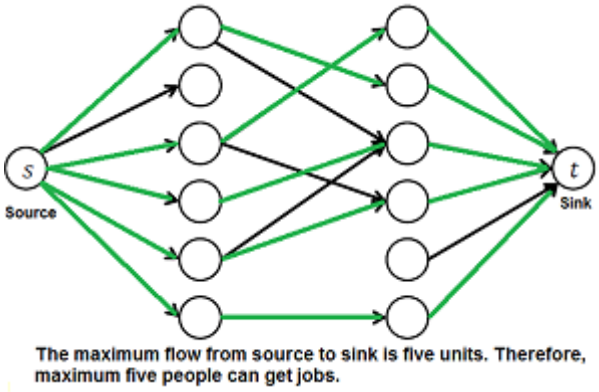
A matching in a **Bipartite Graph** is a set of the edges chosen in such a way that no two edges share an endpoint. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer a matching. There can be more than one maximum matchings for a given Bipartite Graph.

Read about “**Ford-Fulkerson Algorithm for Maximum Flow Problem”** in this link and in Lecture 7: Network Flow

Maximum **B**ipartite **M**atching (**MBP**) problem can be solved by converting it into a flow network (See **this** video to know how did we arrive this conclusion). Following are the steps.



1) Build a Flow Network : There must be a source and sink in a flow network. So we add a source and add edges from source to all applicants. Similarly, add edges from all jobs to sink. The capacity of every edge is marked as 1 unit.



2) Find the maximum flow: We use **Ford-Fulkerson algorithm** to find the maximum flow in the flow network built in step 1. The maximum flow is actually the MBP we are looking for.

Independent Set

In graph theory, an independent set, stable set, coclique or anticlique is **a set of vertices in a graph, no two of which are adjacent**. That is, it is a set of vertices such that for every two vertices in , there is no edge connecting the two. Equivalently, each edge in the graph has at most one endpoint in.

Brute-Force Algorithm

> A brute force algorithm **solves a problem through exhaustion**: it goes through all possible choices until a solution is found. The

> time complexity of a brute force algorithm is often proportional to the input size. Brute force algorithms are simple and consistent, but very slow.

There are no efficient algorithm known for solving this algorithm. The obvious brute-force algorithm would try and check all subsets of nodes to check each one if it is independent. This is possibly the closes we get to solving this problem.

This is a problem that is termed **NP-complete**, which is described more in depth in Lecture 8: NP and Computational Intractability