Module 1 - Algorithm Development

Stages of Algorithm

Session 1 - 5th Jan, 2022

Stages of Algorithm development

Step by Step

- Describe the Problem
- Identifying a suitable technique
- Design of an algorithm
- Proof of Correctness of the algorithm

Stable Matching Problem

Outline

- Students apply for Internship to Companies
- Companies hire Students as Interns
- In both the cases, they have their own preferences
- If a student receives more than one offer, he has a choice to select
- Similarly, a company will have more choice in the selection of Intern
- If any student, denies an offer, it has to be given to the student in next waitlist of the company
- Similarly, If a students receives an offer from the company that he is being waiting for, then he will be forced to say “NO” to the order, currently he is holding.
- So in this scenario, Given a set of preferences among employers and applicants, can we assign applicants to employers so that for every employer E, and every applicant A who is not scheduled to work for E, at least one of the following two things is the case?
  - E prefers every one of its accepted applicants to A; or
  - A prefers her current situation over working for employer E.
- If this holds, the outcome is stable: individual self-interest will prevent any applicant/employer deal from being made behind the scenes.

Formulating the Problem

Step 1

- Each applicant is looking for a single company, but each company is looking for many applicants;
- Initially, to understand, lets eliminate the company from the scene. Each of n applicants applies to each of n companies, and each company wants to accept a single applicant.
- We observe that this special case can be viewed as the problem of devising a system by which each of n men and n women can end up getting married: our problem naturally has the analogue of two “genders”—the applicants and the companies—and in the case we are considering, everyone is seeking to be paired with exactly one individual of the opposite gender.

Matching the Suitable Preference

Not possible all the Time

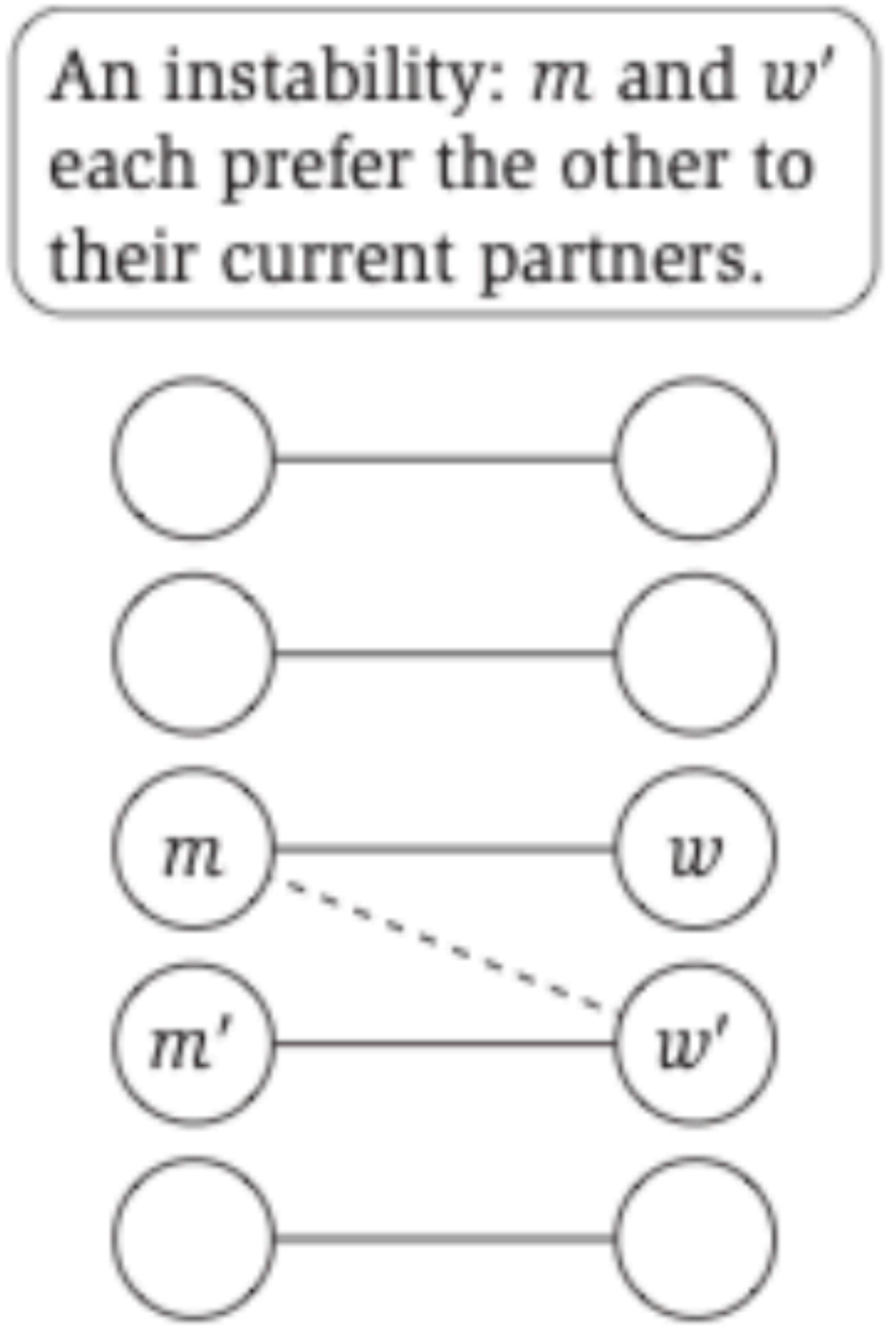

Figure 1.1 Perfect matching S with instability (m, w').

So consider a set $M = \{m_1, \ldots, m_n\}$ of n men, and a set $W = \{w_1, \ldots, w_n\}$ of n women. Let $M \times W$ denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$. A *matching* S is a *set* of ordered pairs, each from $M \times W$, with the property that each member of M and each member of W appears in at most one pair in S. A *perfect matching* S' is a matching with the property that each member of M and each member of W appears in *exactly* one pair in S'.

- In the present situation, a perfect matching corresponds simply to a way of pairing off the men with the women, in such a way that everyone ends up married to somebody, and nobody is married to more than one person—there is neither singlehood nor polygamy.

Instability

A painful process

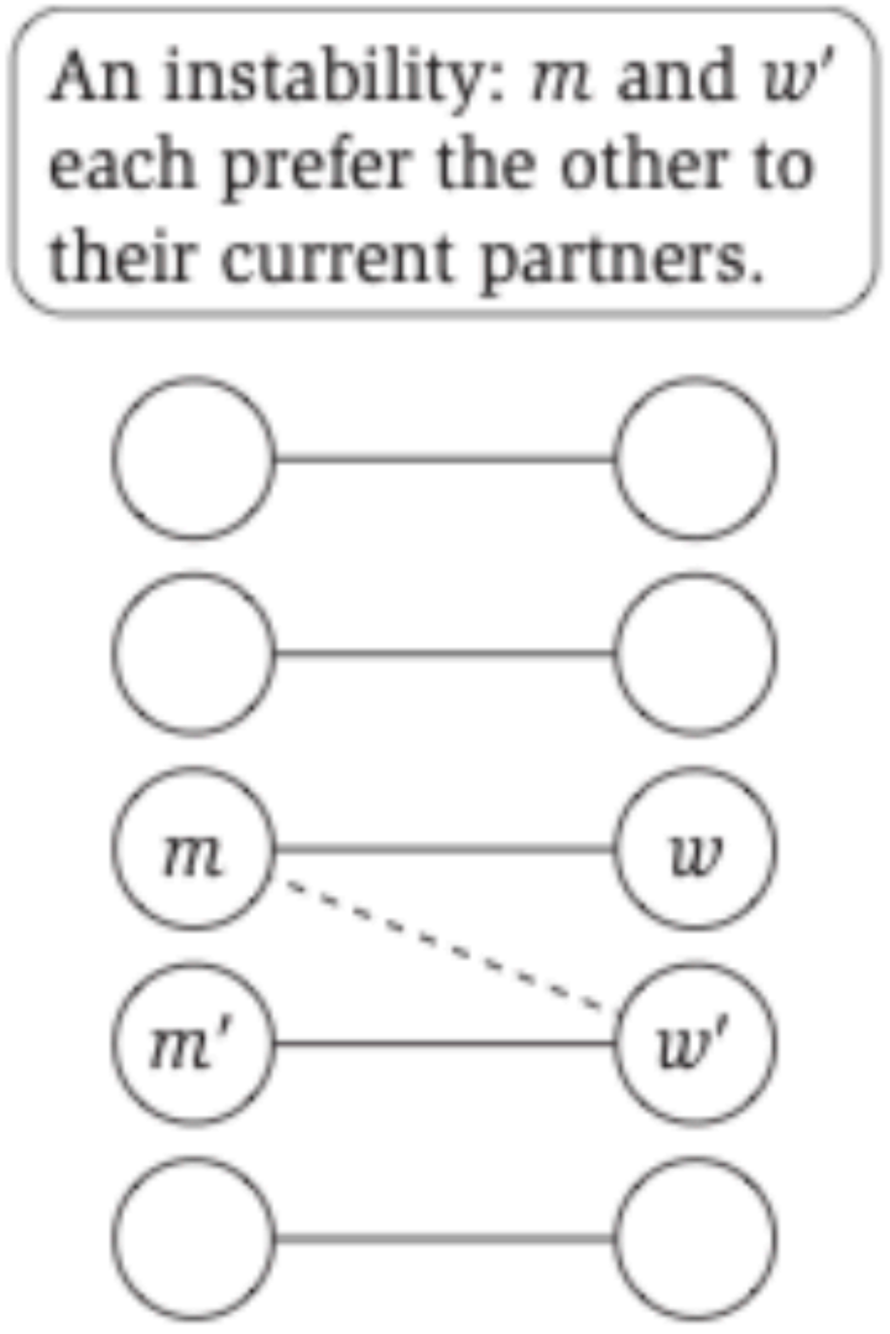

Figure 1.1 Perfect matching S with instability (m, w′).

- Each man m ∈ M ranks all the women; we will say that m prefers w to w′ if m ranks w higher than w′. We will refer to the ordered ranking of m as his preference list. We will not allow ties in the ranking. Each woman, analogously, ranks all the men.
- For a perfect matching, the following situation may arise.
  - There are two pairs (m,w) and (m′,w′) in S with the property that m prefers w′ to w, and w′ prefers m to m′.
- The set of marriages is not self- enforcing. We'll say that such a pair (m, w′) is an instability with respect to S: (m, w′) does not belong to S, but each of m and w′ prefers the other to their partner in S.

Our Goal

Common perspective

- Set of marriages with no instabilities. We'll say that a matching S is stable if (i) it is perfect, and (ii) there is no instability with respect to S.
  - Does there exist a stable matching for every set of preference lists?
  - Given a set of preference lists, can we efficiently construct a stable matching if there is one?
- For Instance,
  - m prefers w to w′.
  - m′ prefers w′ to w.
  - w prefers m′ to m.
  - w′ prefers m to m′.