Algorithms & Complexity: Lecture 6, The Stable Matching Problem

Sam Barrett

March 4, 2021

1 The stable matching problem

Allocation is a fundamental task to life. We want any set of allocations we make to be *stable*. An allocation is stable if there are no unstable pairs, where an unstable pair is a pair that do not *want* to be together, and would prefer a different matching. To achieve this we must introduce a notion of *preferences* .

Allocations can be between 2 individuals from the same set or 2 individuals from different sets. For example Employees $\mapsto$ Teams and Doctors $\mapsto$ Hospitals.

1.1 Matchings within one group

Given an example with 4 people A, B, C and D with the following preferences:

- Preferences for A are $B > C > D$
- Preferences for B are $C > A > D$
- Preferences for C are $A > B > D$
- Preferences for D are $A > B > C$

There are 3 possible matchings:

1. (A, B) and (C, D)
   (B, C) is an **unstable** pair
   This is the case as B prefers C over A and C prefers B over D.
2. (A, C) and (B, D)
   (A, B) is an **unstable** pair
3. (A, D) and (B, C)
   (A, C) is an **unstable** pair

In the above example you can clearly see that no stable matching exists.

1.2 Matchings between two groups

Now we consider the case when we try to allocate between two disjoint sets. Does there always exist a stable matching or, like in the previous case, do there sometimes exist settings where stable matchings do not exist?

Given an example of allocations between hospitals and students, each of size n. Each hospital has a ranking of the n students and each student has a ranking of the n hospitals. We **assume** the list of preferences are strict and complete.

Not all matchings are stable.

- Consider two hospitals h_1, h_2 and two students s_1, s_2.
- Both hospitals prefer s_1 over s_2
- Both students prefer h_1 over h_2
- Therefore, the matching (h_1, s_2) and (h_2, s_1) is not stable as h_1 and s_1 form an unstable pair.

1.3 Definition

Definition 1 *(The `STABLE MATCHING` problem) The `STABLE MATCHING` problem asks to find a stable matching, if one exists.*

The `STABLE MATCHING` problem is in **NP**. This is the case as for n elements, there are $n^2 - n$ pairs in a candidate certificate, each of which is possibly unstable and must be checked individually. If no unstable pair is found then the matching is stable.

The brute force algorithm for `STABLE MATCHING` tries all $n!$ possible matchings, this is very slow as $n! \approx 2^{n \log n}$.

2 Gale-Shapely Algorithm

A better algorithm was proposed by Gale and Shapely in 1962 for complete lists which always finds a stable matching where all elements are allocated. Their algorithm runs in $O(n^2)$ time and puts `STABLE MATCHING` into **P**

The pseudocode for this algorithm is as follows:

Algorithm 1: The Gale-Shapley algorithm (1962)

```
1  Initially all hospitals and students are free
2  while There is a hospital which is free and hasn't made an offer to
     every student do
3  |  Choose such a hospital h
4  |  Let s be highest ranked student to which h hasn't made an offer yet
5  |  if s is free then
6  |  |  (s, h) are matched
7  |  else
8  |  |  s is currently matched to some hospital h'
9  |  |  if s prefers h' to h then
10 |  |  |  h remains free
11 |  |  else
12 |  |  |  s prefers h to their current match h'
13 |  |  |  (s, h) get matched and h' becomes free
14 |  |  end
15 |  end
16 end
```

The running time of this algorithm is clearly $O(n^2)$ as each while loop makes 1 new offer and there can only be n^2 offers made before termination.

2.1 Correctness

From the pseudocode we can see that after receiving their first offer, students always have a better offer or as good an offer *in hand* . We can also see that if a hospital is *free*, then there is a student to whom they have not yet made an offer. Therefore, upon termination, all hospitals and students are matched. But is the matching stable?

Theorem 1 *(Gale-Shapely returns a stable matching)*

- *Let us suppose that it does not, and there therefore exists an unstable pair h and s'*
- *Let (h, s) and (h', s') be allocations in the result. Then we can say that for the unstable pair h, s' to exist h must prefer s' over s and s' must prefer h over h'*
- *The last offer made by h was to s*
- *Did h make an offer to s' before making an offer to s?*
  - *If **NO** then h prefers s to s' **CONTRADICTION***
  - *If **YES** then h was rejected by s' in favour of some other hospital. By our previous point, a student's offers keep getting better*

(or stay the same) and since $h' \neq h$ it follows that s' prefers its final offer h' to h

There is a surprising property of the Gale-Shapely algorithm: it always returns the **SAME** stable matching.

To understand why we require the following definitions:

For each hospital h

- Let $\texttt{Valid}(h) = \{\forall s : S, \text{for which there is a stable matching which matches } h \text{ to } s\}$
- $\texttt{Best}(h)$ is the highest ranked (in preference of h) student from $\texttt{Valid}(h)$

The theorem can be written as "Gale-Shapely always returns the matching with which matches h to $\texttt{Best}(h)$ for each hospital h"

Similarly, we can show that for each student s, s gets their **worst** possible choice.

- Let $\texttt{Valid}(s) = \{\forall h : H, \text{for which there is a stable matching which matches } s \text{ to } h\}$
- $\texttt{Worst}(s)$ is the lowest ranked (in preference of s) hospital from $\texttt{Valid}(s)$

The Gale-Shapely algorithms always returns the matching which matches s to $\texttt{Worst}(s)$ for each student s

You can however, reverse this algorithm in the sense that you can make it so that the students make the offers to the hospitals, this procedure maintains the property that it returns the **same** matching each time but instead of the students getting their worst choices and hospitals getting their best, hospitals get their worst choices and students get their best!

3 Extensions

What about the case in which the two groups are of different size?

Project allocation in the school of Computer Science

- Say we have $5n$ students and n members of staff
- Each faculty member has a preference list over $5n$ students
- Each student has a preference list over n staff
- We want to have a stable allocation of 5 students per faculty member.

Can we extend Gale shapely? **Yes**, this is what is used in MSci project allocation.

The stability of a matching is just the start of desirable properties in a matching. We can add many more conditions, such as *no one gets allocated their n^{th} choice* etc.