Étude 1 – Pairing

Brief:

This report discusses the problem of pairing students together for COSC326 etudes, and an algorithm to make this easy and effective. The idea is that all members of the class can simply fill out a form that lists their own strengths and weaknesses in five different areas of skill, then using an algorithm it can be determined which pairs should be created from the resulting form data, thus resulting in skills balanced between the two people for all formed pairs.

Algorithm:

As our problem is shockingly similar to the "Stable roommates problem", the best solution for this would be to use the algorithm proposed in 1985 by Robert W. Irving[1], which finds a stable matching for an even sized set, where "matching" is defined as "a separation of the set into disjoint pairs"[2], and "stable" is defined as "If there are no two elements which are not roommates and which both prefer each other to their roommate under the matching"[2].

The algorithm has three main phases, and has an exception where a stable solution will not be reached if one person is rejected by all others.

Phase 1: Make Initial proposals

A preference matrix is created from the gathered input data in an order from highest preference to lowest preference going from left to right for each person. Each person then raises a proposal to their highest preference where it may be accepted initially, but if their proposal was superseded by another's proposal later on, then they simply move down their preference list and raise another proposal, and the preference matrix is updated according to whether the proposal was accepted or rejected. This loops until everyone has an agreed proposal.

Phase 1 Example:

<u>Initial phase 1 preference matrix</u>

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference	5th Preference
Arthur	Ben	Cameron	Eva	Francesca	Danielle
Ben	Eva	Danielle	Arthur	Cameron	Francesca
Cameron	Eva	Francesca	Ben	Arthur	Danielle
Danielle	Arthur	Cameron	Francesca	Ben	Eva
Eva	Arthur	Ben	Danielle	Francesca	Cameron
Francesca	Ben	Ralph	Danielle	Cameron	Eva

Proposals are made row by row until everyone has made a proposal that has not been rejected, resulting in a new preference matrix.

<u>Resulting phase 1 preference matrix</u>

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference	5th Preference
Arthur	Ben	Cameron	Eva	~~Francesca~~	~~Danielle~~
Ben	Eva	Danielle	Arthur	Cameron	~~Francesca~~
Cameron	~~Eva~~	Francesca	Ben	Arthur	Danielle
Danielle	~~Arthur~~	Cameron	Francesca	Ben	Eva
Eva	Arthur	Ben	Danielle	Francesca	~~Cameron~~
Francesca	~~Ben~~	~~Ralph~~	Danielle	Cameron	Eva

Phase 2: **Rules out worse matches**

Using our resulting preference matrix from phase 1, for each person in the matrix look at who has proposed to them and cross off each person who is ranked lower than the current proposal. We will then call the resulting matrix our reduced matrix. If any line in the reduced matrix becomes empty of proposals, then a stable solution cannot be found.

Phase 2 Example:

Initial phase 2 preference matrix

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference	5th Preference
Arthur	Ben	Cameron	Eva	~~Francesca~~	~~Danielle~~
Ben	Eva	Danielle	Arthur	Cameron	~~Francesca~~
Cameron	~~Eva~~	Francesca	Ben	Arthur	Danielle
Danielle	~~Arthur~~	Cameron	Francesca	Ben	Eva
Eva	Arthur	Ben	Danielle	Francesca	~~Cameron~~
Francesca	~~Ben~~	~~Ralph~~	Danielle	Cameron	Eva

For the selected candidate, eliminate anyone who is lower on their preference list than the person who has made a proposal to them.

Resulting phase 2 reduced matrix

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference	5th Preference
Arthur	Ben	Cameron	Eva	~~Francesca~~	~~Danielle~~
Ben	Eva	~~Danielle~~	Arthur	~~Cameron~~	~~Francesca~~
Cameron	~~Eva~~	Francesca	~~Ben~~	Arthur	Danielle
Danielle	~~Arthur~~	Cameron	Francesca	~~Ben~~	~~Eva~~
Eva	Arthur	Ben	~~Danielle~~	~~Francesca~~	~~Cameron~~
Francesca	~~Ben~~	~~Ralph~~	Danielle	Cameron	~~Eva~~

Phase 3: **Finalize the best matching**

We now create a 2xn-1 matrix, where n = number of people looking for matchings. The first row is labelled p, and the second is labelled q, and we will call this the finalize matrix.

1. Select a person with two or more proposals left in their row inside the reduced matrix, and they now become the first value in the p row inside the finalize matrix.
2. Looking at the reduced matrix row of the first p value, take their second proposal and make it the first value of the q row inside the finalize matrix.
3. Looking at the reduced matrix row of the first q value, take their last proposal and make it the second value of the p row inside the finalize matrix.
4. Repeat these steps until the same person appears in a row twice inside the finalize matrix, essentially finding a cycle.
5. Once a cycle is found, eliminate more people from the reduced matrix using symmetrically matched values in the finalize matrix, matching the first index of the q row with the second index of the p row, iterating through the finalize matrix sequentially.
6. If the reduced matrix still has more than one proposal in any row, then use the first five steps to find another cycle, and reduce the matrix further until there is only one proposal in each row.

Once step 6 has been completed successfully, the correct configuration has been reached, which means that a stable matching has been found for each person, and the algorithm is complete.

Phase 3 Example:

Initial phase 3 reduced matrix:

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference	5th Preference
Arthur	Ben	Cameron	Eva	~~Francesca~~	~~Danielle~~
Ben	Eva	~~Danielle~~	Arthur	~~Cameron~~	~~Francesca~~
Cameron	~~Eva~~	Francesca	~~Ben~~	Arthur	Danielle
Danielle	~~Arthur~~	Cameron	Francesca	~~Ben~~	~~Eva~~
Eva	Arthur	Ben	~~Danielle~~	~~Francesca~~	~~Cameron~~
Francesca	~~Ben~~	~~Ralph~~	Danielle	Cameron	~~Eva~~

Find the cycle to reduce the matrix even further.

Cycle matrix:

p:	Arthur	Danielle	Cameron	Eva	Arthur
q:	Cameron	Francesca	Arthur	Ben	

Reduce the matrix further using the found cycle.

Resulting phase 3 reduced matrix:

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference	5th Preference
Arthur	~~Ben~~	Cameron	~~Eva~~	~~Francesca~~	~~Danielle~~
Ben	Eva	~~Danielle~~	~~Arthur~~	~~Cameron~~	~~Francesca~~
Cameron	~~Eva~~	~~Francesca~~	~~Ben~~	Arthur	~~Danielle~~
Danielle	~~Arthur~~	~~Cameron~~	Francesca	~~Ben~~	~~Eva~~
Eva	~~Arthur~~	Ben	~~Danielle~~	~~Francesca~~	~~Cameron~~
Francesca	~~Ben~~	~~Ralph~~	Danielle	~~Cameron~~	~~Eva~~

For this example, only one cycle iteration was required to get a desired reduced matrix.

Non-Stable Example:

In this example, all of the candidates have Dan listed as their lowest preference, and , so when Irving's algorithm is applied, because Dan is unanimously disliked, he continually gets rejected, resulting in preference matrix that is unresolvable using phase two, and phase three, thus the algorithm exits at phase 1, because a solution is not possible.

Initial preference matrix:

Candidate	1st Preference	2nd Preference	3rd Preference
Amy	Bob	Cath	Dan
Bob	Cath	Amy	Dan
Cath	Amy	Bob	Dan
Dan	Bob	Cath	Amy

Phase 1 is applied.

Resulting preference matrix:

Candidate	1st Preference	2nd Preference	3rd Preference
Amy	Bob	Cath	~~Dan~~
Bob	Cath	Amy	~~Dan~~
Cath	Amy	Bob	~~Dan~~
Dan	~~Bob~~	~~Cath~~	~~Amy~~

Etude Solution:

Phase 1:

Legend:

	Available
	Updated
	Eliminated

Preference matrix (0):

Candidate	1st Preference	2nd Preference	3rd Preference
1	2	4	3
2	1	3	4
3	1	2	4
4	1	2	3

1. 1 proposes to 2, and gets accepted.
2. 2 proposes to 1, and gets accepted.
3. 3 proposes to 1, and there is a conflict.
4. 1 now chooses between 2 or 3, and as 1 prefers 2, 3 is rejected.
5. The preference matrix in now updated symmetrically.

Preference matrix (1):

Candidate	1st Preference	2nd Preference	3rd Preference
1	2	4	~~3~~
2	1	3	4
3	~~1~~	2	4
4	1	2	3

6. 3 proposes again to 2, and there is conflict.
7. 2 now chooses between 1 or 3, and as 2 prefers 1, 3 is rejected.
8. The preference matrix is now updated symmetrically.

Preference matrix (2):

Candidate	1st Preference	2nd Preference	3rd Preference

1	2	4	~~3~~
2	1	~~3~~	4
3	~~1~~	~~2~~	4
4	1	2	3

9. 3 proposes again to 4, and gets accepted.
10. 4 proposes to 1, and there is a conflict.
11. 1 now choses between 2 or 4, and as 1 prefers 2, 4 is rejected.
12. The preference matrix is now updated symmetrically.

Preference matrix (3):

Candidate	1st Preference	2nd Preference	3rd Preference
1	2	~~4~~	~~3~~
2	1	~~3~~	4
3	~~1~~	~~2~~	4
4	~~1~~	2	3

13. 4 proposes again to 2, and there is a conflict.
14. 2 rejects 4 as they only prefer 1.
15. The preference matrix is now updated symmetrically.

Preference matrix (4):

Candidate	1st Preference	2nd Preference	3rd Preference
1	2	~~4~~	~~3~~
2	1	~~3~~	~~4~~
3	~~1~~	~~2~~	4
4	~~1~~	~~2~~	3

16. 4 proposes again to 3, and gets accepted.
17. The pairings (1,2), and (3,4) are matched and considered stable, so the algorithm is complete.

Phase 2, and phase 3 are not needed as we have already reached a preference matrix with only one preference in each row using only phase 1 of the algorithm.

Bibliography:

[1] *Irving, Robert W. (1985), "An efficient algorithm for the "stable roommates" problem", Journal of Algorithms, **6** (4): 577–595, doi:10.1016/0196-6774(85)90033-1*

[2] https://en.wikipedia.org/wiki/Stable_roommates_problem#CITEREFIrving1985