DSA 5113: Advanced Analytics and Metaheuristics

Homework #2
Instructor: Charles Nicholson

Homework team:
Saeid Hosseinipoor
Reza Alizadeh

February 2017

1 Totally Unimodular Matrices

An integer matrix A is *totally unimodular (*TU*)* if:

1. all entries are -1, 0 or 1
2. at most two non-zero entries appear in any column
3. the rows of A can be partitioned into two disjoint sets such that
    - if a column has two entries of the same sign, their rows are in different sets
    - if a column has two entries of different signs, their rows are in the same set

According to the definition, properties (1) and (2) hold by both given matrices. Therefore; property (3) should be investigated for them which determine whether they are a TU or not.

There is no way to hold property (3) by matrix A_1, because it is not possible to disjoint this matrix such that 1 s separate into two different matrices.

1) $\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{array}$ 2) $\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{array}$ 3) $\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 \end{array}$

Figure 1. Property 3 of TU is not held by Matrix A_1

With respect to the matrix A_2, it is possible to identify two disjoint set – s_1 and s_2 – those can meet the requirements of the 3rd property:

$$\begin{array}{c|c} \begin{array}{ccccccc} -1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array} & s_1 \\ \hline \begin{array}{ccccccc} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{array} & s_2 \end{array}$$

Figure 2. Property 3 of TU is held by Matrix A_2

To test the matrices a MATLAB code is attached to the report which is very useful to verify matrices as a TU.

2 Airport Tractor

The classical shortest path is the best fit for this problem. illustrates the problem in form of a directed graph. Nodes are the representative of the beginning of the year and arcs are the net cost which might be algebraic summation of the various in/out cash flows. The given costs are assumed as follows:

$$C_{ij} = P_i + M_{ij} - T_j \qquad \text{Eq 1}$$

In which, P_i stands for the purchasing price of the tractor in year i, M_{ij} stands for the maintenance cost of the tractor between year i and j and T_j stands for the trading revenue of the tractor in year j. The network flow of the problem is illustrated as follows:

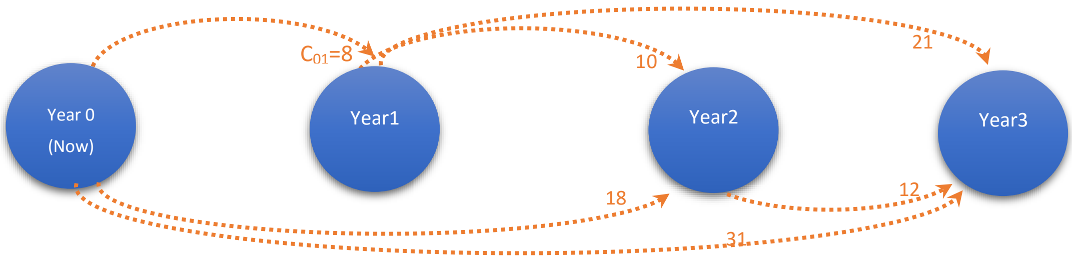

Figure 3. Airport Tractor as a Shortest Path Problem

The objective is to find the shortest (less cost) path from year 0 to year 1. As shown in , there are 4 possible paths from the present year to the 3rd year. All the paths with their costs have been summarized in Table 1.

Table 1. Paths with their costs

Possible path	***Path's cost***	Shortest path in terms of the minimum cost
0-1-2-3	8+10+12=30	×
0-2-3	18+12=30	×
0-3	31	×
0-1-3	8+21=29	✓

The problem modeled by AMPL also resulted same as manual solution as was expected. AMPL code is attached to the report.

3 Christmas Party and Summer Camp

These two problems seem very similar, but they are slightly different. Both could be formulated as linear integer programing. In following problems are formulated and it would be easy to solve if there was an instance of the problem available. They might have more than one possible solution depends on the given information.

3.1 *Christmas Party*

This problem can be solved like an assignment problem with one additional constraint. The participants may describe their like with 1, dislike with -1, and neutral desire as 0. A matrix will show the pairs of desire which acts like favor matrix. The objective is to maximize total favor:

3.1.1 Objective

$$Maximize \sum_{i=1}^{30}\sum_{j=1}^{30} w_{ij}x_{ij} \qquad \text{Eq 2}$$

3.1.2 Constrains

$$Subject\ to:$$

$$\forall x_{ij} \in \{0,1\}, \forall i,j \in \{1,\ldots,30\} \qquad \text{Eq 3}$$

$$x_{ij} = x_{ji}$$

$$x_{ii} = 0$$

$$\sum_{i=1}^{30} x_{ij} = 1$$

$$\sum_{j=1}^{30} x_{ij} = 1$$

In which: w_{ij} is a matrix which represents the like, dislike, and neutrality with 1, -1, and 0, if person i likes to make a pair with person j it is 1 and so on. As apparently, no one can form a pair with itself; therefore, the diagonal would be zero as well. The variable x_{ij} is a binary variable represents the assignment/matching of person i to person j, taking value 1 if the assignment/matching is made and 0 otherwise.

3.2 *Summer Camp*

This problem is a classical stable marriage problem (Wikipedia, n.d.) which solved by (Gale & Shapley, 1962). Here, this problem formulated like the Christmas party problem with small changes. To adopt the previous modeling for this part, some constrains should be added to considered that a French student must be allocated with an English student and vice versa. To make it easier in visualization and formulation, it assumed that the students in the matrix are ordered in such a way that the first 15 students are English and the second 15 ones are French. The another difference of this problem is the weight (desire) matrix which has elements of $\{1, \dots, 10\}$ instead of $\{-1,0,1\}$. Matrix X is divided into four matrices such that:

$$X = \begin{bmatrix} X_{EE} & X_{FE} \\ X_{EF} & X_{FF} \end{bmatrix} \qquad \text{Eq 4}$$

$$X_{EE} = X_{FF} = Zeros = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \qquad \text{Eq 5}$$

$$X_{EF} = X_{FE}^{T} \quad or \quad x_{ef} = x_{fe} \qquad \text{Eq 6}$$

3.3 *Possible Solutions*

According to the proposed formulation, both Christmas party and summer camp problems could be solved by a LP solver as a linear programming problem. Moreover, in both problems, the matrices are TU, because constrains force them to hold all the properties of TU definition (Figure 4).

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, \dots, n \\ & \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1, \dots, n \\ & \mathbf{x} \geq 0 \end{aligned}$$

Figure 4. Why constrain matrices are TU matrices

4 INFORMS Annual Meeting and the Demand for Clean Table Cloths

This problem is a network flow problem in which source and sink are connected to balance the follow. The objective is to minimize the cost of the flow. The schematic flow is shown in Figure 5. The same approach as regular network problem has been conducted. AMPL simulation file is attached to the report for more investigations.

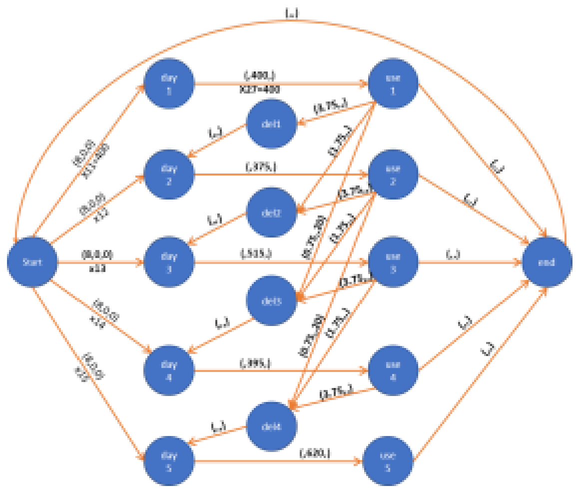

Figure 5. Schematic of INFORMS conference problem

The results are summarized in Table 2 which is associated with total cost of $9251.25.

Table 2. Optimum solution for INFORMS conference problem

Day	***Receive***	***Send***
1	New: 400	CC Laundry: 400
2	New: 375	CC Laundry: 375
3	New: 115 CC Laundry: 400	WW Laundry: 20 CC Laundry: 495
4	WW Laundry: 20 CC Laundry: 375	WW Laundry: 125 Hold: 275
5	CC Laundry: 495 CC Laundry: 125	Hold: 620

5 Assigning Teachers and Classes

This problem is a classic timetabling and scheduling problem which is solved with different approaches. The most important part of the solution is being able to recognize that which professor teaches exactly which courses at what time in which classroom. A solution that does not have certainty about the parameters is incomplete. Refer thesis proposed by (Lovelace, 2010), a good way to model the problem is using the Cartesian product set of related nodes.

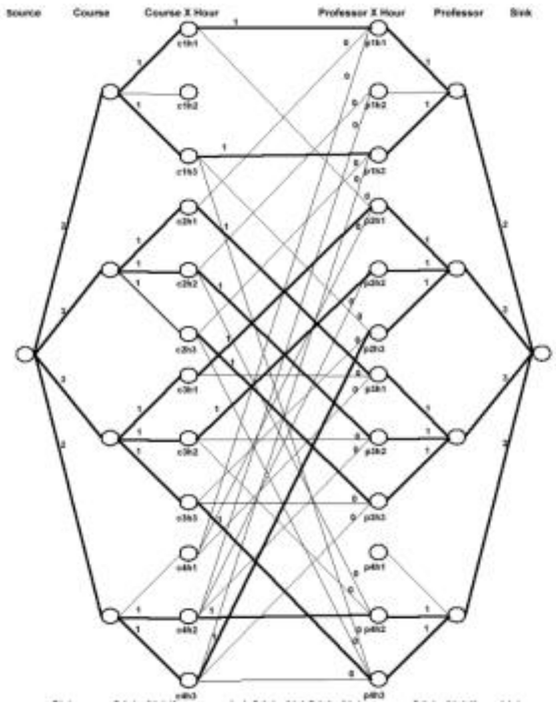

Figure 6. Schematic of typical timetabling and scheduling problem (Lovelace, 2010)

The difference between Lovelace problem and ours is having different constrains and here it is required to define the classroom. Therefore; we have 7 columns of nodes instead of 5. Figure 7 illustrates the problem based on the nodes defined in the model. To simplify the figure, most of the nodes are not drawn where dashed arrows address those missing nodes. The real model is more complicated than this picture.

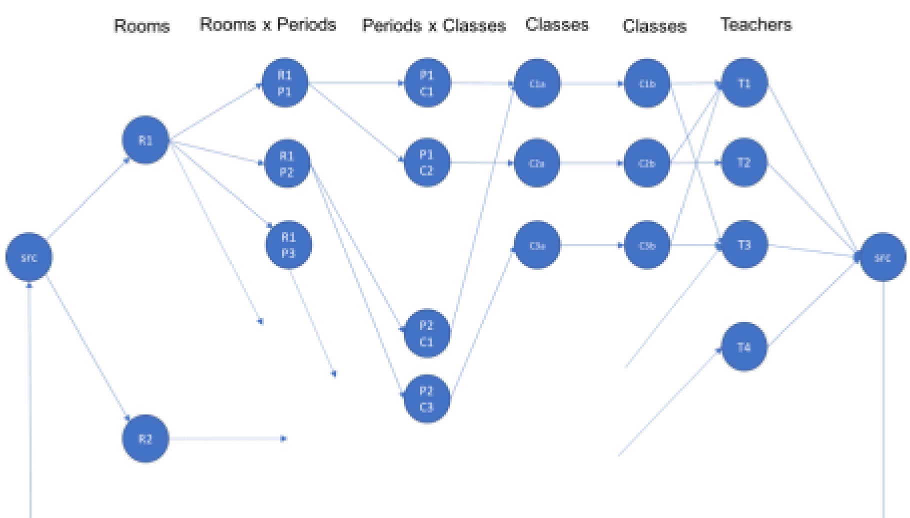

Figure 7. Simplified diagram for timetabling and scheduling problem

After imposing constrains on the graph and solving the problem as a flow network, the following result could be obtained.

Table 3. Result for scheduling problem

Class	***Teacher***	***Period***	***Room***
1	1	4	1
2	2	3	2
3	1	3	1
4	1	2	2
5	3	2	1
6	2	1	2
7	3	3	1
8	2	6	2

The problem has more than one solution. As can be seen on Table 3, teacher 4 is not assigned to any classes. To have a better solution, it is suggested to add a new constrain that each teacher at least should teach one class. Table 4 shows the new results with the new added constrain. For more details please refer to the AMPL model and solution which is attached to this report.

Table 4. Result for scheduling problem (better solution after adding a new constrain)

Class	***Teacher***	***Period***	***Room***
1	3	4	1
2	2	3	2
3	3	3	1
4	1	2	2
5	3	2	1
6	4	1	2
7	4	1	1
8	4	6	2

6 Boomer Global Air Services

There is too many information in the problem statement which are trivial and non-important. Figure 8 shows the necessary data mined from the cumbersome and messed-up descriptions.

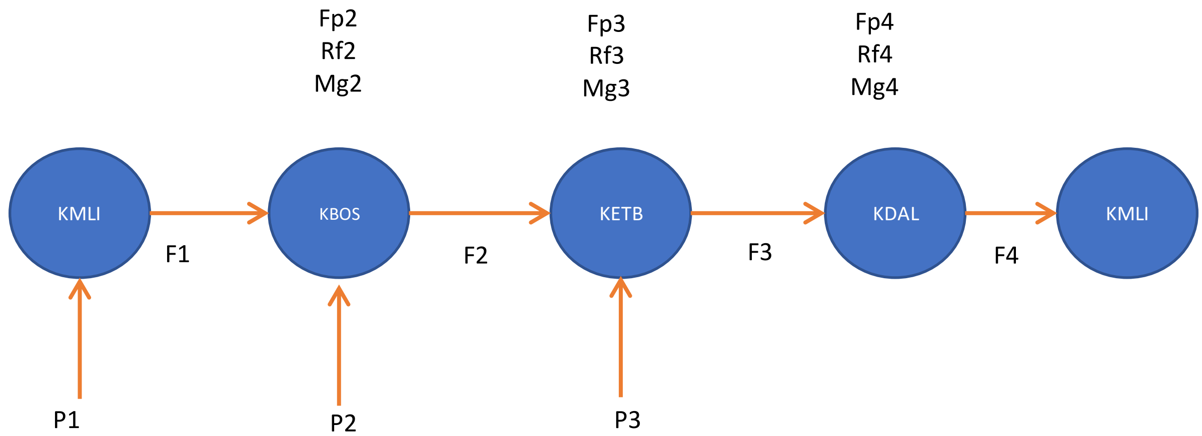

Figure 8. Schematic for fuel plan problem

***Note:**

Fpi: Fuel price ($/gallon)	Fp1= 3.97, Fp2=8.35, Fp3=7.47, Fp4
Rfi : Ramp fee in station i ($)	Rf1=0, rf2= 800, Rf3=450, Rf4=400
Mgi: Minimum gallons to waive fee	Mg1=0, Mg2=500, Mg3=300, Mg4=350
Pi: The number of passengers	P1=2, P2=2, P3=4
Fi: Fuel burn incl. taxi (pounds)	F1=4800, F2=2000, F3=5300, F4=3100

6.1 *Assumptions*

1. The given data for burnt fuel is related to CE750.
2. Based on the fuel consumption rates, the burnt fuel for GV was calculated as 1.5 time of CE750's.
3. As mentioned vaguely in the problem statement, it was assumed that airplane is ready with minimum amount of fuel in its tank and at the end of journey, the tank should be refilled up to the minimum amount of fuel level.

According to these assumptions data for both CE750 and GV airplanes are available. There is no information about the type of airplane that should fly on the mentioned trip. Obviously, the cost related to CE750 should be much less than GV, but the problem modeled in general form and was expected it chose CE750 for the trips. Just by small changes to the data file, we can add more airplane to the model.

6.2 *Objective*

Objective is to minimize the cost of fuels based on the constrains.

6.3 *Constrains*

There are many constrains from regulations like minimum fuel amount on landing, maximum weight of airplane in landing and take-off as well as some definitions for variables like the relationship between fuel consumptions and the fuel in tank, capacity of tank and so on. The following constrains are directly copied from AMPL model file which is attached to this report for more investigations:

```
subject to Fuelling {i in Airport}:                  # Filling in airport i
        fTank[i] = eTank[i] + Fuel[i];

subject to Tank_Capacity {i in Airport}:     # Tank Capacity Constrain
```

```
        fTank[i] <= sum {j in Airplane} cTank[j] * x[j];

subject to Safety_Stock {i in Airport}:       # Filling in airport i
        eTank[i] >= sum {j in Airplane} eTanklim[j] * x[j];

subject to Readiness:
        eTank['KMLIi'] = sum {j in Airplane} hTank[j] * x[j];
subject to Readiness_f:
        fTank['KMLIf'] >= sum {j in Airplane} hTank[j] * x[j];

subject to Fuel_Ramp_Fee {i in Airport}:  # Ramp fee constrain
        z[i] <= (Fuel[i] / (6.7 * RFlim[i])); #+ M * zz[i];

subject to Fuel_Cost {i in Airport}:                      # Fuel cost calculation
        Cost[i] = RFee[i] * (1 - z[i]) + c[i] * Fuel[i] / 6.7;

subject to Cost_limit {i in Airport}:          # Fuel cost constarin
        Fuel[i] <= M * y[i];

subject to Landing_Weight_KMLIi:                  # Landing Weight Calculation
        WL['KMLIi'] = eTank['KMLIi'] + sum {j in Airplane} BOW[j] * x[j] + 200 * 0;
subject to Landing_Weight_KBOS:                           # Landing Weight Calculation
        WL['KBOS'] = eTank['KBOS'] + sum {j in Airplane} BOW[j] * x[j] + 200 * 2;
subject to Landing_Weight_KTEB:                           # Landing Weight Calculation
        WL['KTEB'] = eTank['KTEB'] + sum {j in Airplane} BOW[j] * x[j] + 200 * 2;
subject to Landing_Weight_KDAL:                           # Landing Weight Calculation
        WL['KDAL'] = eTank['KDAL'] + sum {j in Airplane} BOW[j] * x[j] + 200 * 4;
subject to Landing_Weight_KMLIf:                  # Landing Weight Calculation
        WL['KMLIf'] = eTank['KMLIf'] + sum {j in Airplane} BOW[j] * x[j] + 200 * 0;

subject to Ramping_Weight {i in Airport}: # Ramping Weight Calculation
        WR[i] = WL[i] + Fuel[i] + 200 * Pass[i];

subject to Max_Land_Weight {i in Airport}:        # Maximum Landing Weight Constrain
        WL[i] <= sum {j in Airplane} MWL[j] * x[j];

subject to Max_Ramp_Weight {i in Airport}:        # Maximum Ramping Weight Constrain
        WR[i] <= sum {j in Airplane} MWR[j] * x[j];

subject to Fligh_Fuel_Consumption_1:              # Flight Fuel Consumtion
        eTank['KBOS'] = fTank['KMLIi'] - sum {j in Airplane} FCons['KMLIi',j] * x[j];
subject to Fligh_Fuel_Consumption_2:              # Flight Fuel Consumtion
        eTank['KTEB'] = fTank['KBOS'] - sum {j in Airplane} FCons['KBOS',j] * x[j];
subject to Fligh_Fuel_Consumption_3:              # Flight Fuel Consumtion
        eTank['KDAL'] = fTank['KTEB'] - sum {j in Airplane} FCons['KTEB',j] * x[j];
subject to Fligh_Fuel_Consumption_4:              # Flight Fuel Consumtion
        eTank['KMLIf'] = fTank['KDAL'] - sum {j in Airplane} FCons['KDAL',j] * x[j];

subject to Select_Airplane:                                       # Choose airplane
        sum {j in Airplane} x[j] = 1;
```

6.4 *Optimum Solution (part a)*

The result for optimum solution is summarized in Table 5. According to this solution, even by paying fee in Boston, it is more affordable to neglect the fee waiver respect to filling the expensive fuel. The total cost for this plan is ***$11645.2***.

Table 5. Optimum solution for fuel plan problem

Airport	***Tank Level @ arrival (lb)***	***Fueling (lb)***	***Tank Level @ departure (lb)***	***Cost ($)***
KMLIi	7000	6000	13000	3555.22
KBOS	8200	0	8200	800.00
KTEB	6200	2010	8210	2241.00
KDAL	2910	2590	5500	2323.27
KMLIf	2400	4600	7000	2725.67

6.4.1 No Tankering Solution

No tinkering means avoid filling up the tank in the home airport which has the cheapest fuel but fuel up to the amount of tank level that satisfies the minimum tank level for safety. The following constrain should be added to the model:

```
subject to No_Tankering:
        eTank['KBOS'] = sum {j in Airplane} eTanklim[j] * x[j];
```

Results are summarized in Table 6. Total cost in this case increased to ***$14205.9***.

Table 6. No tankering solution for fuel plan problem

Airport	***Tank Level @ arrival (lb)***	***Fueling (lb)***	***Tank Level @ departure (lb)***	***Cost ($)***
KMLIi	7000	200	7200	118.51
KBOS	2400	3350	5750	4175.00
KTEB	3750	3950	7700	4403.96
KDAL	2400	3100	5500	2780.75
KMLIf	2400	4600	7000	2725.67

The most important limitation in the model is ***fuel consumption calculation*** which is considered as very rough values for each airplane in individual flight. Fuel consumption is a function of airplane weight that is consisting of variable weight of passenger and fuel itself. This variable part is not negligible and might be third of the total weight.

The other limitation is the ***fuel constant price***. In real world, the fuel price is not constant and varied by time that should be updated every time. It is possible, in some situation, providers grant discount for certain amount of fuel purchases. It could be addressed by a stepwise function for fuel price instead of fixed price.

Passengers and their luggage's weight could be entered as parameter not a constant value. It is not such important because the deviation from real values is very small compare to total weight of airplane.

Weather condition also affects the fuel consumption that could be entered in fuel consumption calculations.

6.5 *100 Gallons Constrain (part b)*

To define and satisfy this constrain, we need to add two constrains with M such that:

```
var zz {Airport}  binary;                    # Dummy binary for fuel purchase adjustment


subject to Part_b1 {i in Airport}:
		Fuel[i] <= M * zz[i];
subject to Part_b2 {i in Airport}:
		Fuel[i] + M * (1-zz[i]) >= 100 * 6.7;
```

Table 7 summarizes the results which as exactly same as optimum solution results, because the airplane doesn't take any fuel but greater than 100 gallons (670 lbs).

Table 7. Solution with 100-gallon constrain for fuel plan problem

Airport	***Tank Level @ arrival (lb)***	***Fueling (lb)***	***Tank Level @ departure (lb)***	***Cost ($)***
KMLIi	7000	6000	13000	3555.22
KBOS	8200	0	8200	800.00
KTEB	6200	2010	8210	2241.00
KDAL	2910	2590	5500	2323.27
KMLIf	2400	4600	7000	2725.67

7 References

Fourer, R., Gay, D. M. & Kernighan, B., 1993. *Ampl.* Danvers(MA): Boyd & Fraser.

Gale, D. & Shapley, L., 1962. College Admissions and the Stability of Marriage. *The American Mathematical Monthly,* 69(1), pp. 9-15.

Lovelace, A. L., 2010. *On the complexity of scheduling university courses..* [Online] Available at: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1255&context=theses

Wikipedia, n.d. *Wikipedia.* [Online] Available at: https://en.wikipedia.org/wiki/Stable_marriage_problem