OpenRules Decision Model for
DMCommunity June-2024 Challenge "Smart Marriages"
(Rule Solver)

Problem

This challenge deals with the famous stable marriage problem formulated as follows:

"Given n men and n women, where each person has ranked all members of the opposite sex in order of preference, marry the men and women together such that there are no two people of the opposite sex who would both rather have each other than their current partners. When there are no such pairs of people, the set of marriages is deemed stable."

Here is an example of the problem:

How Men Rank Women:

	Alice	Barbara	Claire	Doris	Elsie
Adam	5	1	2	4	3
Bob	4	1	3	2	5
Charlie	5	3	2	4	1
Dave	1	5	4	3	2
Edgar	4	3	2	1	5

How Women Rank Men:

	Adam	Bob	Charlie	Dave	Edgar
Alice	1	2	4	3	5
Barbara	3	5	1	2	4
Claire	5	4	2	1	3
Doris	1	4	3	2	5
Elsie	4	2	3	5	1

Approach

Instead of complex marriage stability constraints, we will implement simple constraints that state:

"Each person should marry one and only one person of the opposite gender".

Then we define preferences for all pairs "man-woman" and "woman-man" and calculated the sum of all these preferences. It became our optimization objective. When we minimize this objective, we receive a solution with the most top preferences being satisfied.

Business Part

We start with the creation of a business glossary:

Glossary glossary				
Variables	**Business Concept**	**Attribute**	**Type**	**Used As**
Men	**Marriages**	men	Person[]	in
Women		women	Person[]	in
Name	**Person**	name	String	
Preferences		preferences	int[]	
Assigned Pairs	**Result**	assignedPairs	String[]	out

Then we create test data for two instances of the problem of sizes 5 and 6:

DecisionData Person men5	
Person Name	**Person Preferences**
Adam	5,1,2,4,3
Bob	4,1,3,2,5
Charlie	5,3,2,4,1
Dave	1,5,4,3,2
Edgar	4,3,2,1,5

DecisionData Person women5	
Person Name	**Person Preferences**
Alice	1,2,4,3,5
Barbara	3,5,1,2,4
Claire	5,4,2,1,3
Doris	1,4,3,2,5
Elsie	4,2,3,5,1

DecisionData Person men6	
Person Name	**Person Preferences**
Adam	1,2,4,5,6,3
Bob	4,2,5,3,1,6
Charlie	2,5,4,6,3,1
Dave	3,4,2,6,5,1
Edgar	6,3,2,1,5,4
Fred	4,6,3,2,1,5

DecisionData Person women6	
Person Name	**Person Preferences**
Alice	5,2,6,3,1,4
Barbara	6,5,1,2,4,3
Claire	3,2,1,4,5,6
Doris	2,4,1,3,5,6
Elsie	3,5,4,2,1,6
Fiona	5,1,6,4,2,3

We add these samples as test cases for my OpenRules-based decision model and added expected results from my previous solution:

DecisionTest testCases			
#	Define	Define	Expect
Test	**Women**	**Men**	**Assigned Pairs**
1	women5	men5	Adam-Barbara Bob-Doris Charlie-Elsie Dave-Alice Edgar-Claire
2	women6	men6	Adam-Alice Bob-Barbara Charlie-Claire Dave-Fiona Edgar-Doris Fred-Elsie

To check that our test data satisfies the requirement “there should be the same number of men and women”, we add the following decision table:

Decision ValidateData			
Condition		Message	ActionExecute
Count of Men		ERROR	Rules
!=	Count of Women	"Number of men and women should be the same"	ACTION-TERMINATE

Solver Part

I used the following table to create all unknown decision variables, one for each pair man_woman:

Decision DefineAllVariables [for each Man in Men; for each Woman in Women]	
SolverCreate	
Variable Name	**Domain**
{{Name of Man}}_{{Name of Woman}}	0-1

They all are constrained variables with possible values of 0 or 1 and names like “Adam_Alice”, “Adam_Barbara”, etc.

To state that *“each man should marry one and only one woman”* I decided to organize a loop for each Man in the array of Men:

Decision PostMenConstraints [for each Man in Men]
ActionExecute
Decision Tables
DefineManVariables
PostManConstraints

This loop executes the following decision tables:

Decision DefineManVariables [for each Woman in Women]	
SolverAddVariableToArray	
Array Name	**Var Expression**
{{Name of Man}}Variables	{{Name of Man}}_{{Name of Woman}}

Decision PostManConstraints
SolverPost
Constraint Expression
Sum({{Name of Man}}Variables) = 1

Similarly, we create 3 tables to state that *"each woman should marry one and only one man"*. Here they are:

Decision PostWomenConstraints [for each Woman in Women]
ActionExecute
Decision Tables
DefineWomanVariables
PostWomanConstraints

Decision DefineWomanVariables [for each Man in Men]	
SolverAddVariableToArray	
Array Name	**Var Expression**
{{Name of Woman}}Variables	{{Name of Man}}_{{Name of Woman}}

Decision PostWomanConstraints
SolverPost
Constraint Expression
Sum({{Name of Woman}}Variables) = 1

To define an array “AllPreferences” we organized two loops, one for men and one for women:

Decision DefineManPreferenceVariables [for each Person in Men]	
SolverAddVariableToArray	
Array Name	**Var Expression**
AllPreferences	{{Name of Person}}Variables * Preferences

Decision DefineWomanPreferenceVariables [for each Person in Women]	
SolverAddVariableToArray	
Array Name	**Var Expression**
AllPreferences	{{Name of Person}}Variables * Preferences

For each Person from the array Men suvh as Adam it adds a scalar product of Adam Variables and Adam Preferences to the array “AllPreferences”. The second table does the same for all Women.

Now we can specify the problem objective as a sum of all veraible in the array “AllPreferences”:

Decision SetObjective
SolverSetObjective
Objective
sum(AllPreferences)

Finally, as we usually do for Rule Solver, I needed to define two methods “Define” and “Solve” to complete my decision model. Here is the method “Define” that simply executes previously specified decision tables:

Decision Define
ActionExecute
Decision Tables
ValidateData
DefineAllVariables
PostMenConstraints
PostWomenConstraints
DefineManPreferenceVariables
DefineWomanPreferenceVariables

And here is the method “Solve” that sets our optimization objective “AllPreferences” and uses the standard methods “Solver Minimize” and “SolverLogSolution” to minimize and print the optimal solution:

Decision Solve
ActionExecute
Actions
SetObjective
SolverMinimize
SolverLogSolution
AssignSolution

To save the found solution in the array “Assigned Pairs” defined in our Glossary, we can use the following decision table:

Decision AssignSolution [for each Man in Men; for each Woman in Women]				
SolverSolutionCondition			Action	
Variable Name	Operator	Value	Assigned Pairs	
{{Name of Man}}_{{Name of Woman}}	=	1	Add	{{Name of Man}}-{{Name of Woman}}

When I executed my decision model I received the expected results for both test cases that correspond to the same results from a similar implementation but done in Java:

Test 1: Assigned Pairs: [Adam-Barbara, Bob-Doris, Charlie-Elsie, Dave-Alice, Edgar-Claire]

Test 2: Assigned Pairs: [Adam-Alice, Bob-Barbara, Charlie-Claire, Dave-Fiona, Edgar-Doris, Fred-Elsie]

It correspond expectations from the test cases:

DecisionTest testCases			
#	Define	Define	Expect
Test	Women	Men	Assigned Pairs
1	women5	men5	Adam-Barbara Bob-Doris Charlie-Elsie Dave-Alice Edgar-Claire
2	women6	men6	Adam-Alice Bob-Barbara Charlie-Claire Dave-Fiona Edgar-Doris Fred-Elsie

The execution time very fast (under a second).