Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A subsidiary organ of OIC

CSE 4618: Artificial Intelligence

Lab Report 03

Name	**:**	**M M Nazmul Hossain**
Student ID	**:**	**200042118**
Semester	**:**	**6th**
Academic Year	**:**	**2022-2023**
Date of Submission	**:**	**27.03.2024**

Introduction

The Lab involved solving Constraint Satisfaction Problems using a new software CSP-Applet from Alspace.

Problem Analysis

Problem 1- Eating Out:

The problem described a few friends going out to have dinner and were indecisive about what they wanted to order. There were 4 friends and the menu had 4 menu items to choose from. Each person had different preferences which will be further discussed in the explanation section.

Problem 2 - Finding Houses:

The problem described a few people trying to rent an apartment in a building. There were 4 people and 3 floors, and multiple people could live on a single floor. However, there were some constraints which had to be met before assigning each person their floors and apartment. The constraints are further discussed in the explanation section.

Problem 3 - Spots:

The problem described the order in which 6 people could stand in a queue. Since it is a queue, only one person could occupy one spot. They had a few requirements to stand in the queue however, unlike a regular problem, not everyone wanted to be first. The requirements are discussed in the explanation section below.

Problem 4 - Scheduling Tasks:

The most difficult problem by far, bar none to implement in this lab. It described the schedule of two teachers. They had to accomplish 5 tasks for which both teachers had only 4 time slots each. So, in total, there were 8 time slots combining both teachers and the 5 tasks needed a total of 7 slots since two of the tasks needed 2 times slots to be completed. One of the

slots does get filled in by one of the constraints, all of which will be further discussed in the explanation section below.

Solution and Explanation

Problem 1- Eating Out:

The preferences were listed as this person wanted this, or didn't want this. So, in order to keep things simple, the variables were chosen as the 4 friends and what they could order was their domain.

The constraints were set according to the preferences of each person:

Unary Constraints:

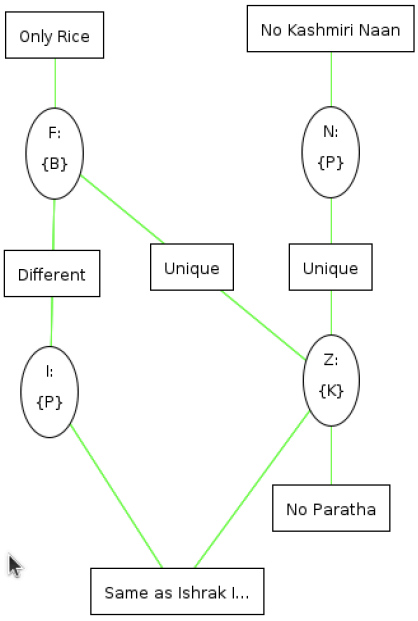
Farabi (F) could only order Rice items (S and B) and couldn't order (K or P), Zahid (Z) couldn't order Parathas (P) since he doesn't like them and Nafisa (N) couldn't order Kashmiri Naan (K) since she had them before already.

Binary Constraints:

Ishrak (I) and Farabi (F) couldn't order the same item since they wanted to share their food, Zahid (Z) had to order food differently from Nafisa (N) and Farabi (F) since he wanted to be unique. I treated it as 2 separate Binary Constraints instead of a single Trinary Constraint.If possible Zahid (Z) will order the same as Ishrak (I) since he likes copying him. I took the liberty of deciding something different from the specified question here where Zahid wants to order the same item as Ishrak. This means if Zahid can order the same item as Ishrak, he will otherwise order whatever available item that meets all the other constraints. This is because the constraint should be mainly followed by Zahid. It really shouldn't logically limit Ishrak from ordering parathas just because Zahid doesn't like parathas.

These were all the preferences and the preferences met via specified constraints which are pretty self explanatory. The applet provided only True, False, Crossword and Custom options for String variables, which is unfortunate, as it left with no choice other than using the

custom which allowed specifying True for the different domains or a combination of domains for multi variables.



	Z	N	F	I
1	B	S	S	B
2	B	S	S	P
3	B	P	S	B
4	B	P	S	P
5	K	S	S	K
6	K	B	S	K
7	K	P	S	K
8	K	S	S	P
9	K	B	S	P
10	K	P	S	P
11	S	B	B	S
12	S	B	B	P
13	S	P	B	S
14	S	P	B	P
15	K	S	B	K
16	K	B	B	K
17	K	P	B	K
18	K	S	B	P
19	K	B	B	P
20	K	P	B	P

Running Auto Solve on the CSP resulted in it automatically determining a possible solution. Clicking on auto solve multiple times, resulted in it showing all of the possible solutions, none of which would break any of the constraints. In total there were 20 different solutions.

Variables : F,N,I,Z

Domain: {B,S,K,P}

Constraints:

i) F!={K} and F!={P}

ii) Z!={P}

iii)N!={K}

iv)F!=I

v)F!=Z

vi)N!=Z

vii)I==Z if Ishrak doesn't order Parathas, Zahid can order anything else he wants otherwise. These are all custom constraints.

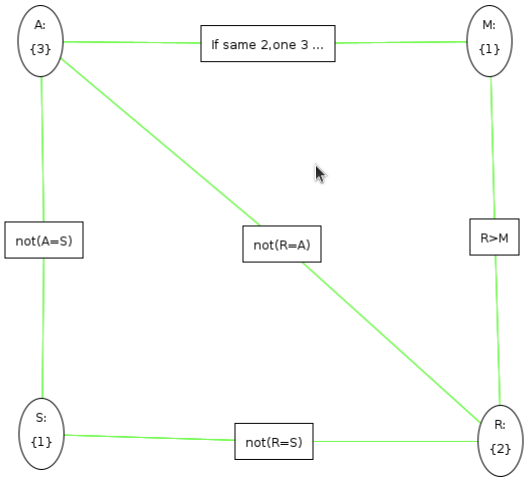
Problem 2 - Finding Houses:

The requirements were listed in the way as X and Y shouldn't live on the same floor, or X and Y can live on the same floor but it must be this. Also since the floors were integer values, it would be easier to design on the Applet. For this reason, the variables were chosen to be the 4 people Ali (A), Sristy (S), Rafid (R), and Maliha (M) and the domains were taken as the 3 floors {1,2,3}. All the constraints here are Binary Constraints.

Binary Constraints:

So Ali and Sristy cannot live on the same floor. Rafid cannot live on the same floor as anyone else. This could be turned into a constraint involving all 4 variables. But for simplicity's sake, I considered it as 3 separate constraints. And Rafid always had to live on a higher floor than Maliha, which replaced one of those 3 constraints as greater automatically means not equal to. And Finally, A and M had to be on floor 2 if they were living on the same floor or one of them had to live on the third floor.

These were all the requirements that had to be met while choosing the apartments. This time around, the domain were integers which came with some fantastic benefits. Namely, the equal to or not equal to settings. I didn't have to set a custom constraint anymore and this was much simpler. During the lab I didn't know this setting existed while doing this task, and discovered it during task 3. Later however, I changed the constraint configuration.



1	A=2	S=1	M=2	R=3
2	A=3	S=1	M=1	R=2

Running Auto Solve on the CSP resulted in it automatically determining a possible solution. Clicking on auto solve multiple times, resulted in it showing all of the possible solutions, none of which would break any of the constraints. In total there were 2 different solutions.

Variables: A,S,M,R

Domains: {1,2,3}

Constraints:

i) A!=S

ii) R!=A

iii)R!=S

iv)R>M

v) A and M need to be on the second floor if they're both on the same floor or one of them has to be on the 3rd floor. Which is a custom constraint.

Problem 3 - Spots:

Since the requirements were all specified as X should be next to Y or X wants to be before last etc, the variables for this problem were chosen to be the 6 people. The variables were Rifat (R), Atiq (A), Farhan (F), Ishmam (I), Tabassum (T) and Sabrina (S). The domain for these variables were the 6 spots. Since This problem had 6 variables, it was a little difficult to ensure all constraints. An All Diff constraint, if it were easy to apply in the applet, would've made life much simpler. However, that didn't seem feasible so an alternative solution had to be found. This time the domains were all integers, so it was possible to use the equal to or not equal to constraints.

Unary Constraints:

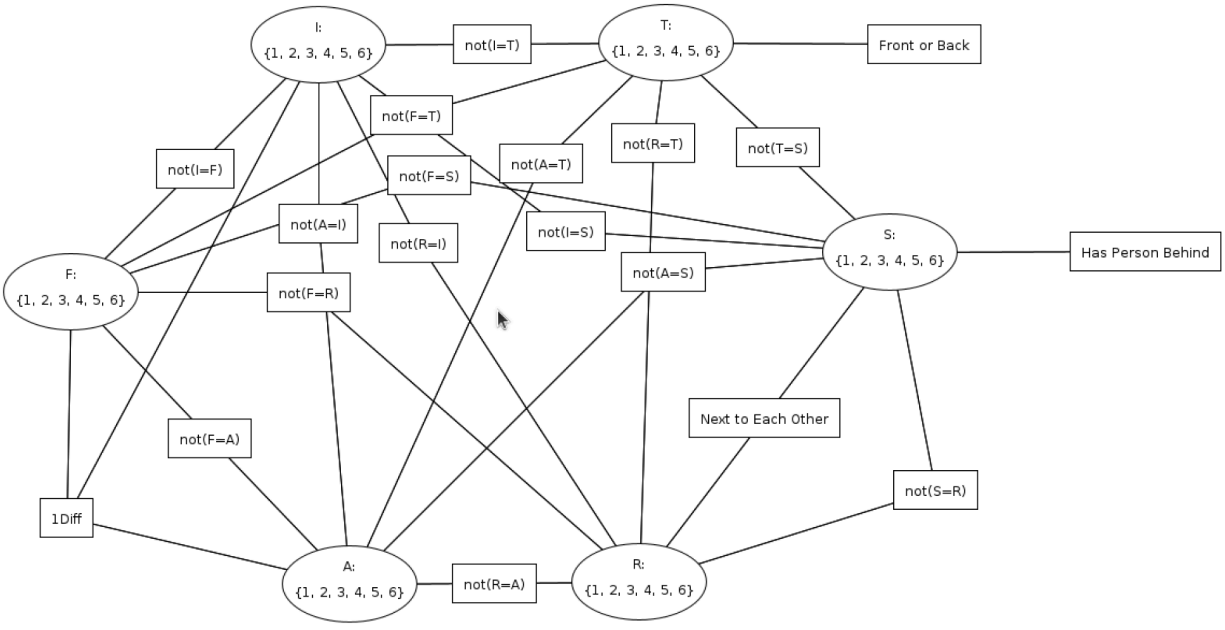
T could only be at either the front of the line or at the very back so only the 1st and 6th spot. S wanted to have at least one person behind her so she couldn't occupy the 6th spot.

Binary Constraints:

S and R wanted to stand next to each other, so no matter where S stood, R had to be either in front of her or behind her. An All Diff constraint, if available here would have done wonders. However, it wasn't feasible as there would be 46656 different boxes. I didn't want to spend the rest of my life just clicking away. I made a mesh of Binary constraints Where R!=A, R!=F, R!=I,R!=S,R!=T, A!=F, A!=I… and so on to achieve a similar effect.

Trinary Constraint:

This next requirement involves three variables A, F, I and multiple requirements. F had to stand between A and I. So, no matter where they stood, A and I would have had to have a difference of 1 between them where F could stand. A could be in front or at the back. According to sir's advice it was assumed that the three friends would always stand together.



Running Auto Solve on the CSP resulted in it automatically determining a possible solution. Clicking on auto solve multiple times, resulted in it showing all of the possible solutions, none of which would break any of the constraints. In total there were 14 different solutions.

	1	2	3	4	5	6
1	T	I	F	A	S	R
2	T	A	F	I	S	R
3	T	R	S	I	F	A
4	T	S	R	I	F	A
5	T	R	S	A	F	I
6	T	S	R	A	F	I
7	I	F	A	R	S	T
8	I	F	A	S	R	T
9	A	F	I	R	S	T
10	A	F	I	S	R	T
11	R	S	I	F	A	T
12	S	R	I	F	A	T
13	R	S	A	F	I	T
14	S	R	A	F	I	T

Variables: R, A, F, I , S, T

Domain: {1,2,3,4,5,6}

Constraints:

i) S!={6}

ii)T={1,6}

iii)S and T are next to each other. This constraint was custom.

iv) A, F, I are next to each other and F is always in the middle. This was also custom.

v) All Diff (R!=A, R!=F,R!=I, R!=S, R!=T, A!=F, A!=I, A!=S, A!=T, F!=I, F!=S, F!=T, I!=S, I!=T, S!=T) which is 15 binary constraints by itself.

Problem 4 - Scheduling Tasks:

This problem constraints involved 5 tasks and 4 time slots of 2 different teachers. To simplify the problem according to the constraints provided, I divided the variables as:
X_8 (8am to 9am), X_9 (9 am to 10 am), X_10 (10 am to 11 am), X_11 (11 am to 12 pm), Y_8 (8am to 9am), Y_9 (9 am to 10 am), Y_10 (10 am to 11 am), and finally Y_11 (11 am to 12 pm) which is a total of 8 variables.

There were 5 tasks, 2 of which required 2 consecutive hours, and one constraint required one teacher to just wait while another did the DBMS Lab. So, the domains for each variable were:
{G,Q1,Q2,C,D,L1,L2,-}.

Q1 and Q2 represented the 2 consecutive hours it took to check the quiz scripts and L1 and L2 represented the 2 consecutive hours it took to complete the AI lab. The - represents one of the faculties attending the DBMS lab.

Due to the immense amount of constraints I had to put in place I won't formally distinguish each one.

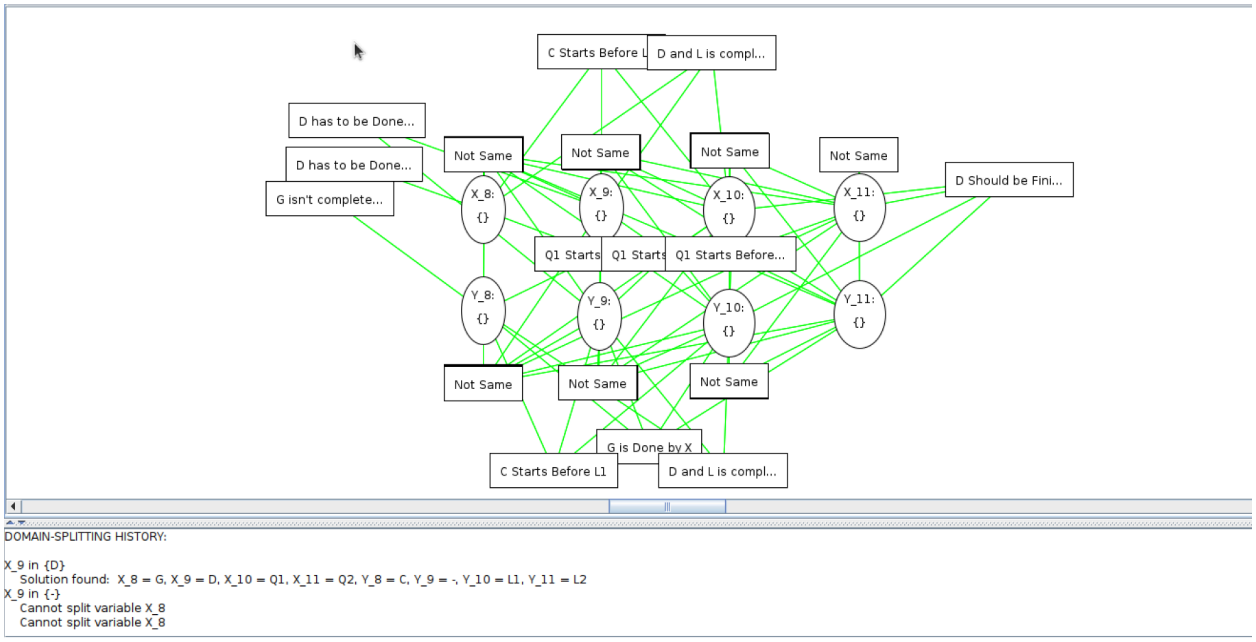
An All Diff if possible would've been a blessing. However, it just wasn't possible and I didn't want to click 8^8 = 16777216 different boxes. This involved String variables as well, so the applet didn't allow me to just give X!=Y. I had to type all of them Manually. I instead opted to follow the mesh method which involved writing (7+6+5+4+3+2+1) = 28 different constraints to achieve the same result as an All Diff.

Then I only considered X8, X9, and X10 or Y8,Y9,Y10 when ensuring that D and L1 are not completed by the same teacher which involved 2 separate Trinary Constraints for X and Y. Similarly only X8, X9, and X10 or Y8,Y9,Y10 were considered when ensuring C is completed before the L1 can take place over 2 separate trinary Constraints for X and Y. I also included the shortcut constraint here as if a teacher completes C, only then, they can attempt to do L1. Otherwise, the same effect could be ensured using a mesh of 28 constraints. To Ensure L1- L2

and Q1-Q2 take place consecutively, I used binary constraints. It ensured that L1 started before L2 and L2 took place immediately after L1 and similarly for Q1 Q2. This required 6 different Binary constraints for Q and L each. So a total of 12 constraints involving X8-X9, X9-X10,X10-X11 and Y8-Y9, Y9-Y10,Y10-Y11.

Then a few more constraints were put in place to ensure that G is not completed by Y since X is good at it. G had to be collected by 8 so x8 had to be G since DBMS lab had to be completed by 10. And since G had to be completed the earliest D could be completed by either X9 or Y9 which was also set as a constraint. If X was taking the lab Y would be attending it and vice versa. Since D should be finished by 10, X10,X11,Y10,Y11 couldn't do D, which was also a constraint.

Around 49 Different constraints were used when building this CSP solution all of which seems a lot of needless work.



Surprisingly, All the Constraints Lead to just 1 Solution. Which is the one mentioned in the screen shot. X8 = G, X9 = D, X10 = Q1, X11 = Q2, Y8 = C, Y9 = -. Y10 = L1, Y11 = L2.

Variables: X_8 (8am to 9am), X_9 (9 am to 10 am), X_10 (10 am to 11 am), X_11 (11 am to 12 pm), Y_8 (8am to 9am), Y_9 (9 am to 10 am), Y_10 (10 am to 11 am), and finally Y_11 (11 am to 12 pm)

Domain: {G,Q1,Q2.C,D,L1,L2,-}

Constraints:

Please don't ask me to write them all down… I will give you the file if you want… Please have some mercy on my poor soul….

Findings and Insights

Problem 1- Eating Out:

	Z	N	F	I
1	B	S	S	B
2	B	S	S	P
3	B	P	S	B
4	B	P	S	P
5	K	S	S	K
6	K	B	S	K
7	K	P	S	K
8	K	S	S	P
9	K	B	S	P
10	K	P	S	P
11	S	B	B	S
12	S	B	B	P
13	S	P	B	S
14	S	P	B	P
15	K	S	B	K
16	K	B	B	K
17	K	P	B	K
18	K	S	B	P
19	K	B	B	P
20	K	P	B	P

In 10 of those solutions, Zahid (Z) was able to order the same food as Ishrak (I) because the other 10 times, Ishrak (I) ordered Parathas(P) which Zahid didn't want to order. Ishrak ordered non Rice items in more solutions, because he wanted to be different from Farabi who only wanted to order rice. Zahid (Z) ended up ordering the Kashmiri Naan (K) in most of the solutions since neither Farabi (F) nor Nafisa (N) wanted to order it and Zahid wanted to be unique. According to my solution, Ishrak (I) isn't restricted from ordering Parathas. But if it was

changed to what the question specified, Ishrak wouldn't be able to order Parathas and there would be only 10 solutions.

	Z	N	F	I
1	B	S	S	B
2	B	P	S	B
3	K	S	S	K
4	K	B	S	K
5	K	P	S	K
6	S	B	B	S
7	S	P	B	S
8	K	S	B	K
9	K	B	B	K
10	K	P	B	K

Problem 2 - Finding Houses:

1	A=2	S=1	M=2	R=3
2	A=3	S=1	M=1	R=2

Among the 2 Solutions, it is noticeable that R never chooses the first floor, since he has to be different from everyone else and he must live on a floor higher than M. It is also noticed that M never gets to live on the 3rd floor because of that same constraint since if M lives on the 3rd floor, there isn't a higher floor left for R. For this reason A can only occupy either 3 or 2 and M can occupy either 1 or 2. And since S doesn't want to be on the same floor as either A or R, S is stuck on the 1st floor.

Problem 3 - Spots:

	1	2	3	4	5	6
1	T	I	F	A	S	R
2	T	A	F	I	S	R
3	T	R	S	I	F	A
4	T	S	R	I	F	A
5	T	R	S	A	F	I
6	T	S	R	A	F	I
7	I	F	A	R	S	T
8	I	F	A	S	R	T
9	A	F	I	R	S	T
10	A	F	I	S	R	T
11	R	S	I	F	A	T
12	S	R	I	F	A	T
13	R	S	A	F	I	T
14	S	R	A	F	I	T

After analyzing the solution, we can see that there is a pattern T can only be at the front or back, which is what happens. S and R want to be next to each other so this pattern (SR and RS) alternates throughout the solution, except in the section 1-2, where SR was seen but RS was never seen. This was because of the constraint that S always wants at least someone behind her, so she couldn't occupy the 6th spot. A similar pattern of IFA and AFI can also be seen throughout the entire solution. The combination of all of this resulted in a total of 14 answers. During the lab we were completely misinformed that there is only one solution which will also follow an alphabetical order. A F I R S T is seen to be a possible solution however. During the lab we weren't informed of the way to find out all the multiple possible answers. Also the settings where arc consistency checking took 1 seconds to complete for each arc would make finding all these solutions almost impossible since there are around 19 different constraints. Setting that to 0s made the process much faster and much more feasible. Using which all the possible solutions to the provided problem were discovered.

Problem 4 - Scheduling Tasks:

This problem kind of broke me. I am so glad to have been done with this. The solution was found to be X8 = G, X9 = D, X10 = Q1, X11 = Q2, Y8 = C, Y9 = -. Y10 = L1, Y11 = L2.

This should've been quite obvious honestly, since DBMS had to be completed by 10 it had to have started by 8 or by 9. And since G can only be completed by X and it couldn't be completed by anyone else, it should be the first task done by X at X8. During that time, Y shouldn't just sit around. It could do one of two things. D isn't possible, neither is L since C hasn't been done yet. The only possible actions were Q1 or C. But, D had to be completed by 10 so it started at 9. Y would have to either attend it or take the class themselves. So if they started Q1 at Y8, Y9 wouldn't be free for attending or D. So the only possible value at Y8 was C. Since Y8 was C Y was now obligated to take the AI Lab L. Som Y could not take the DBMS Lab D. So Y would attend the lab at 9. So X9 = D and Y9 = -. Finally Since Y10 and Y11 would be

taking the Lab L1 and L2, X10 and X11 would be spent checking the quiz script. This is the only possible solution to the provided problem according to all the restrictions.

Other than findings from the solutions, during the lab, it wasn't presented properly to us that clicking on the autosolve multiple times would result in showing multiple solutions to the CSP. Also there was an option that sped up the arc consistency checking if set to 0s in the auto solver, which made this process much faster.

Challenges Faced

While solving the 4th problem, I found out how backdated the applet was to handle this specific problem. If I made any small mistakes like I added a variable to the constraint by mistake. If i wanted to just remove that, the applet would reset all the modifications I had made to the constraint and reset it to all True. I had to redo all of my modifications. I also found out about copy and pasting in the xml file to copy my modifications between constraints, otherwise, this simply would not have been possible.

I hadn't paid too much attention during class, so during this lab, I had very little idea about Constraint Satisfaction Problems. Luckily, the lab was very simple and guides to solving the problems were provided in Google Classroom Past Problems. Using those and the class lecture slides, I had brushed up my knowledge on CSP and tackled the lab. The applet was a completely new technology which I had difficulty grasping full knowledge about during the lab. After returning home and tinkering with it more, I understood how a lot of the features function.

I still do not understand how to properly utilize the crossword.

Additional Information

I liked the fact that the question involved a lot of our seniors. It gave me a chuckle. I also noticed each question involved seniors from incremental years. Following that pattern. Next to next year, there should be some people from our batch who have a question designed after

them. This lab was also very important for me as it helped my understanding of CSP, which honestly helped my preparation for the Mid Semester Examination. What I learned most from this is that constraints are very difficult to implement although most of it was because of technical limitations.