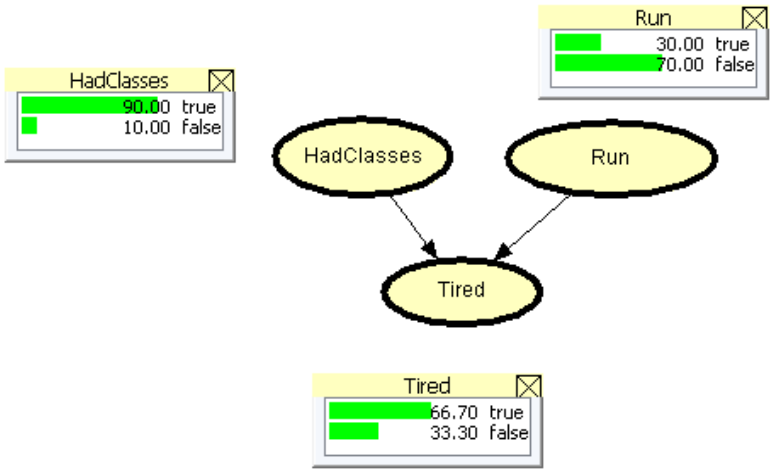
Saúl Abimael Gómez Martínez A01700790

# Laboratory 4

## Example




**Conditional tables:**

**HadClasses**

| | |
|---|---|
| **True** | 0.9 |
| **False** | 0.1 |

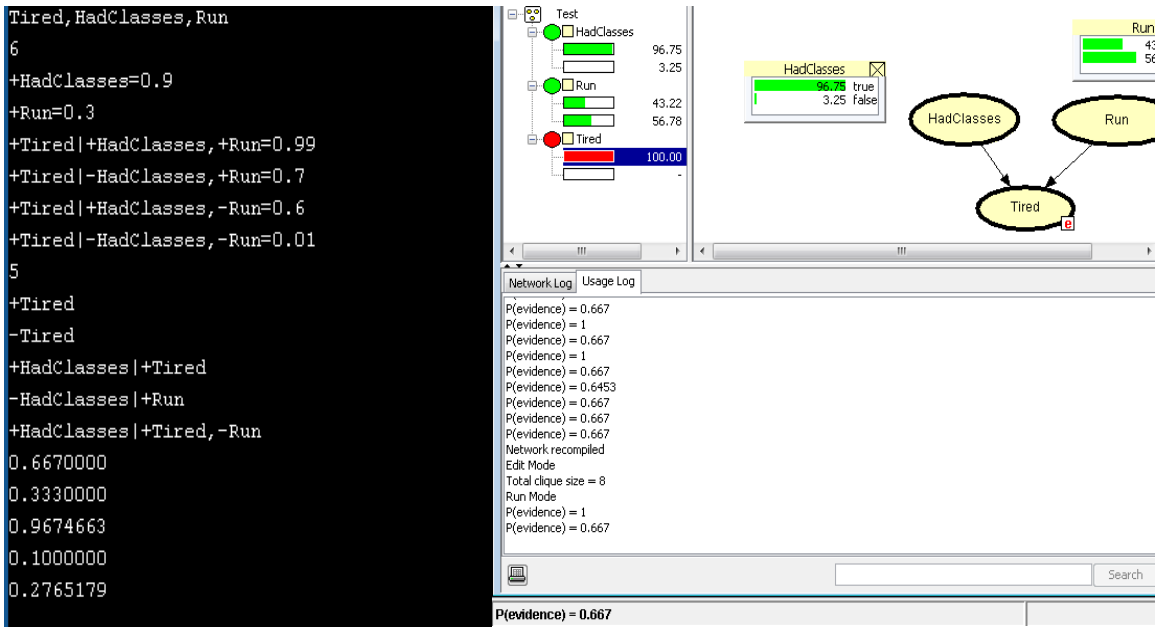
**Run**

| | |
|---|---|
| **True** | 0.3 |
| **False** | 0.7 |

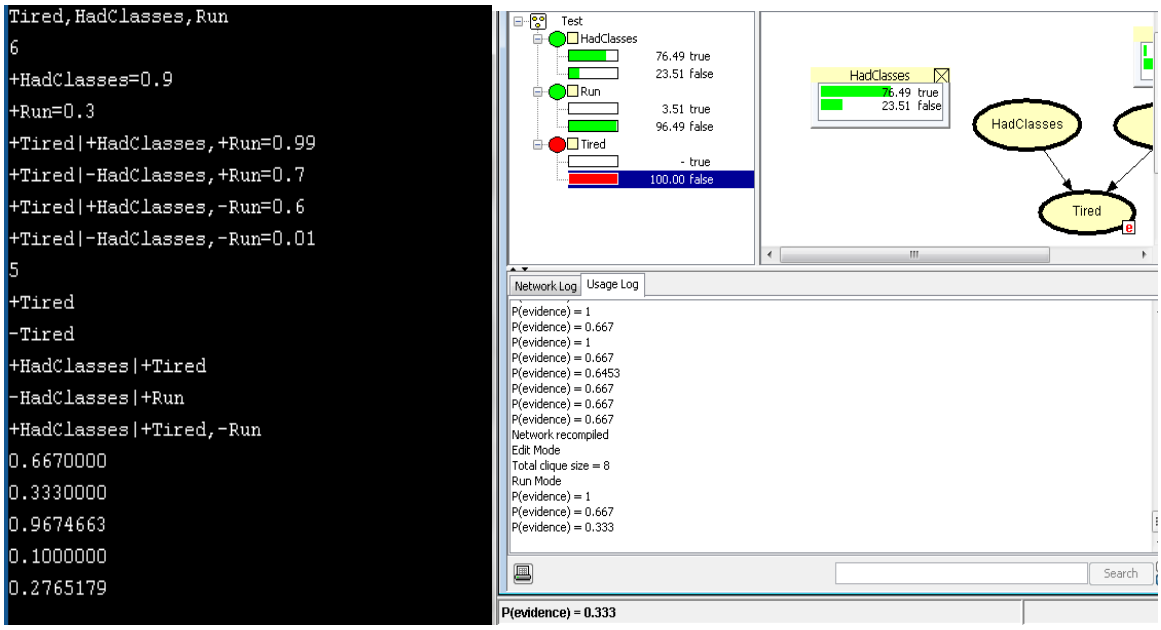
**Tired**

| **Run** | True | | False | |
|---|---|---|---|---|
| **HadClasses** | True | False | True | False |
| **True** | 0.99 | 0.7 | 0.6 | 0.01 |
| **False** | 0.01 | 0.3 | 0.4 | 0.99 |

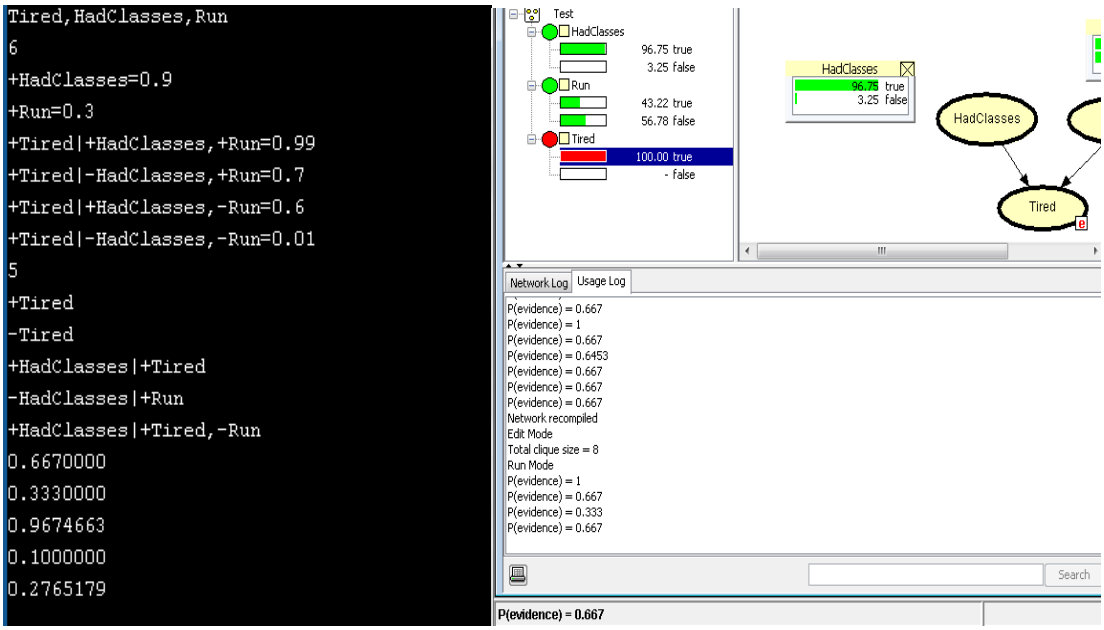
**Comparison with between Hugin and my program**



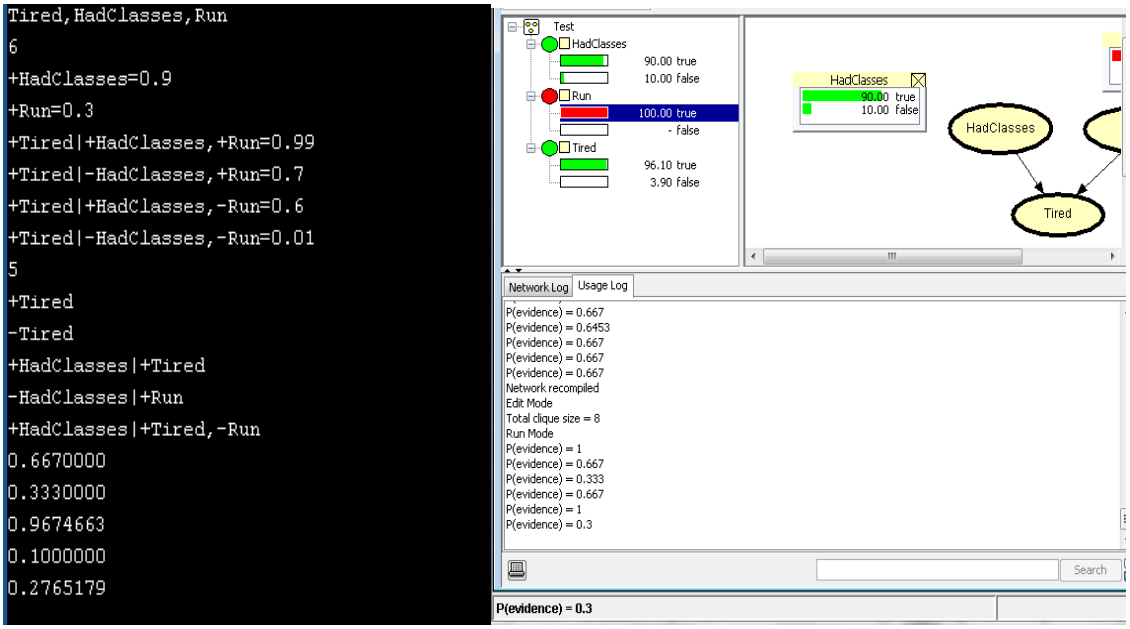

In Hugin, when Tired is true, P(evidence) is equal to 0.667. We get the same value in the program.



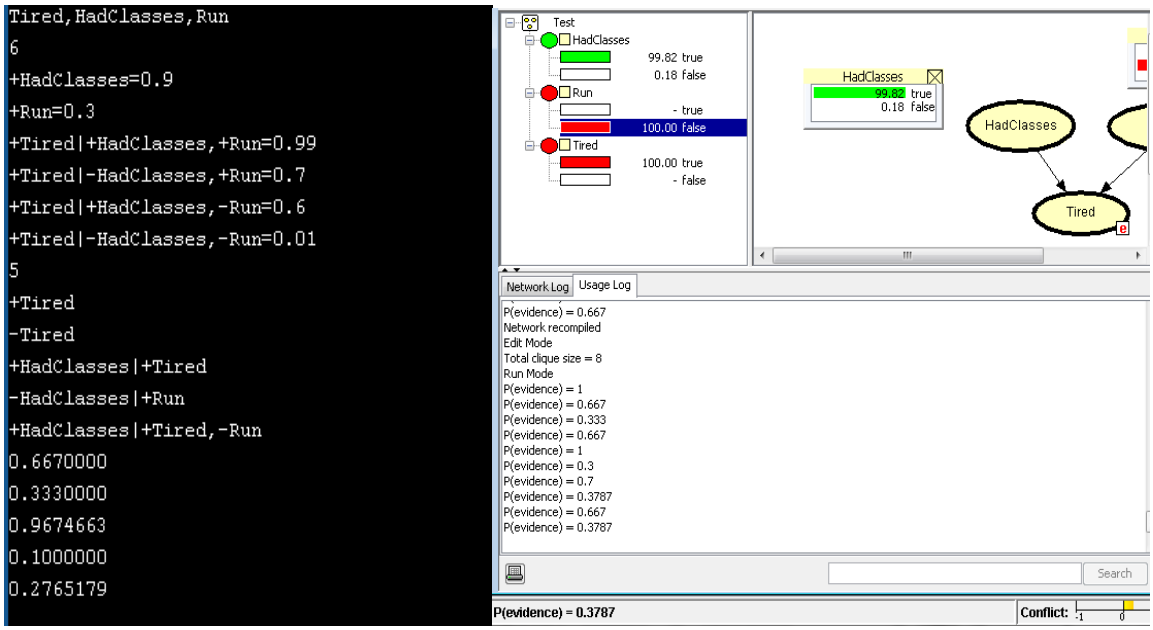

In Hugin, when Tired is false, P(evidence) is equal to 0.333. We get the same value in the program.




In Hugin, given Tired is true the probability in HadClasses for True changes to 96.75. We get the same value in my program.




In Hugin, given Run is True it doesn't have any effect for the case True of HadClasses it remains 0.1. This happens because both are independent of each other. We get the same in value in the program.




In Hugin, given Run is False and Tired is true, the probability for True in HadClasses changes to 0.9982. Unfortunately, we don't get the same value in the program. This could happen because of an error in the calculations or evaluations.

## Reflection

After using the software Hugin Lite I could say it is very useful for calculating and evaluating different conditions for a great variety of situations. In comparison with my program I could say that work very similarly for at least a small case of 5 nodes. But I would need to say that Hugin works much better than my program since my implementation is not able of solving a very complex connection of nodes.

Hugin relates the probabilities and conditions through a graphic and friendly interface. It stablishes the parents and children nodes through its arrow which also interrelate the conditions. When you give it evidence it will print out the value of it, also when one probability is stablished as true or false it will sometimes affect the values of other conditions. This depends in if they are conditionally dependent or independent of each other.

In some sense, both methods ask the user to form its own network, giving name to their nodes, their probabilities, and the relations. The difference with mine is that it does it through typing the name, their symbols for true or false, and their corresponding probability. The problem with this is that if by any chance there is a mistake while typing the input and isn't fixed then it could cause miscalculation or ignore some states.

Despite this possible problem, the biggest advantage of using my program is that it gives you back the condition you ask while in Hugin, in some cases, you need to find it by stablishing the conditions asked. For this aspect I'd say my program is faster for solving some problems.

You can add as many states you want to your nodes in Hugin., not just true or false. This is very helpful for evaluating situations where there are not only 3 answers. Unfortunately, can only be

applied for a true or false situation. If something other than this was given to the program it could give wrong values and states.

Both Hugin and my code calculate different values and conditions in case they are needed this is made so they can evaluate future conditions. My code has a certain level of difficulty, but I believe it still requires a lot more of work so it can work perfectly almost as the Hugin software.

I would only use my code for simple application because of its low capacity to solve complex problems, maybe networks with only 5 nodes for now until the code is corrected. Hugin for now is much better for analyzing a great variety of conditions, like the calculating the probability of raining in a certain day by checking conditions like weather, day, month, season, etc.