**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

| **Module Title:** | Artificial Intelligence  Data Visualisation & Comms |
| --- | --- |
| **Assessment Title:** | AI\_DV \_Lv8\_ICA\_v5 |
| **Lecturer Name:** | David McQuaid  Sam Weiss |
| **Student Full Name:** | Mateus Fonseca Campos |
| **Student Number:** | 2023327 |
| **Assessment Due Date:** | 05/01/2024 |
| **Date of Submission:** | 10/01/2024 |

**Declaration**

| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |
| --- |



|  | CSP: Frameworks, Algorithms and Visualizations |
| --- | --- |
|  | Integrated Continuous Assessment |

# 

| Mateus Fonseca Campos |
| --- |
| 2023327 |

# 

# Contents

[**Tables 3**](#_wp6trhpkjhcf)

[**Figures 4**](#_2zy5yb27m6dw)

[**Introduction 5**](#_30j0zll)

[**Tasks for Artificial Intelligence 6**](#_3znysh7)

[Question 1 6](#_rqb0g4uyzhld)

[Question 2 8](#_f43pwg7barjb)

[Question 3 8](#_rai1zap15z6l)

[**Tasks for Data Visualisation 10**](#_599sls6913be)

[Question 4 10](#_pf2oit6faxsq)

[Question 5 11](#_wv8acvfn7re)

[Question 6 11](#_gydztbpzkd8m)

[Question 7 13](#_d0yg7as6tqzv)

[**Conclusion 14**](#_1ksv4uv)

[**References 15**](#_2jxsxqh)

# Tables

[Table 1: Assignment scheme for variables, value domains and constraints in both Scenarios 1 and 2. 8](#_b7helvamo7oz)

# Figures

[Figure 1: Recursive backtracking algorithm implemented in Python. 10](#_ya1jtxvad22u)

[Figure 2: Scenario 1’s truth table processed by the CSP framework. Note the number of rows. 11](#_smw7uww1jahe)

[Figure 3: Scenario 2’s truth table processed by the CSP framework. Once again, note the number of rows. 12](#_yozoutinxhn9)

[Figure 4: As the number of variables increases incrementally, the number of rows in the table increases exponentially. 12](#_7ll0ei45hpr5)

[Figure 5: Screenshots of Tkinter CSP Application.. 14](#_tlopxga986qo)

# Introduction

This is an integrated assignment whose aim is to bring together the contents discussed during completion of both Introduction to Artificial Intelligence and Data Visualization and Communication modules, more specifically, the concept of Constraint Satisfaction Problems (CSPs). The main focus is to explore tackling the challenge of how to solve such problems, as well as communicate the properties of the applicable scenarios and their potential respective solutions.

Two scenarios are presented in this assessment. In both scenarios a company representative wishes to hire new employees to assume certain roles within the organization. Choosing from a list of candidates, and their respective skills, the goal is to evaluate all possible combinations of candidates that would satisfy the company’s needs while respecting some pre-established constraints, such as funding.

This work is divided into two major sections: Tasks for Artificial Intelligence and Tasks for Data Visualization. In the first part, Questions 1 and 2 discuss the use of CSP frameworks to find possible solutions, their characteristics and main disadvantages, while Question 3 suggests an algorithmic approach as a more efficient alternative. In the second part, Question 4 and 5 deal with communicating the properties of the scenarios, while Question 6 offers a GUI application that allows users to explore the scenarios’ properties in real time. Finally, Question 7 seeks to justifiy the visualization techniques chosen in the previous steps.

# Tasks for Artificial Intelligence

## Question 1

Using the CSP framework python-constraint (Niemeyer, Celles and Willemsen, 2023), both Scenarios 1 and 2 could be solved. In both cases, the variables are the roles and the value domain is the candidates, while the constraints limit which values each role can assume that satisfy the problem. Duplicated roles are treated as different variables. The table below details the assignment scheme:

| **Scenario #** | **Variable** | **Domain** | **Constraint** | **Rationale** |
| --- | --- | --- | --- | --- |
| 1 | Python 1 | All\* | Must be Ciara | Ciara is guaranteed in the team since she is the owner of the company. |
| Python 2 | All\* | Must be Peter, Jane or Bruce | Anyone who knows Python, except Ciara. |
| AI 1 | All\* | Must be Peter, Juan, Jim or Anita | Anyone who knows AI. |
| AI 2 | All\* | Must be Peter, Juan, Jim or Anita | Anyone who knows AI. |
| Web | All\* | Must be Juan, Mary or Anita | Anyone who knows Web. |
| Database | All\* | Must be Jane | In general, anyone who knows Database, but in this case, there is only Jane. |
| Systems | All\* | Must be Jim, Mary or Bruce | Anyone who knows Systems. |
| AI 1 and AI 2 | All\* | Must be different from one another | The two AI roles cannot be performed by the same person. |
| All\*\* | All\* | There must be less than 5 different values across all variables | Team size has to be up to 4, due to funds limitation. |
| 2 | Python 1 | All\* | Must be Ciara | Ciara is guaranteed in the team since she is one of the owners of the company. |
| Python 2 | All\* | Must be Peter, Jane or Bruce | Anyone who knows Python, except Ciara. |
| AI 1 | All\* | Must be Juan | Juan is guaranteed in the team since he is one of the owners of the company. |
| AI 2 | All\* | Must be Peter, Jim or Anita | Anyone who knows AI, except Juan. |
| AI 3 | All\* | Must be Peter, Jim or Anita | Anyone who knows AI, except Juan. |
| Web | All\* | Must be Juan | Juan is guaranteed in the team since he is one of the owners of the company. |
| Database | All\* | Must be Jane | In general, anyone who knows Database, but in this case, there is only Jane. |
| Systems | All\* | Must be Jim, Mary or Bruce | Anyone who knows Systems. |
| AI 2 and AI 3 | All\* | Must be different from one another | The two available AI roles cannot be performed by the same person. |
| All\*\* | All\* | There must be less than 7 different values across all variables | Team size has to be up to 6, due to funds limitation. |

\*Applies to all values in the domain: Ciara, Peter, Juan, Jim, Jane, Mary, Bruce and Anita.

\*\*Applies to all variables in the scenario. Scenario 1: Python 1, Python 2, AI 1, AI 2, Web, Database, Systems. Scenario 2: scenario 1 + AI 3.

### Table 1: Assignment scheme for variables, value domains and constraints in both Scenarios 1 and 2.

After executing the script with the configuration above, the following results were obtained:

* Scenario 1:
  + Number of solutions found: 4.
  + Number of hirees per solution: 4/4.
  + Example:
    - Python: Ciara and Jane.
    - AI: Anita and Jim.
    - Web: Anita.
    - Database: Jane.
    - Systems: Jim.
* Scenario 2:
  + Number of solutions found: 42.
  + Number of hirees per solution: 5/6 and 6/36.
  + Example with 5 hirees:
    - Python: Ciara and Jane.
    - AI: Juan, Anita and Jim.
    - Web: Juan.
    - Database: Jane.
    - Systems: Jim.
  + Example with 6 hirees:
    - Python: Ciara and Bruce.
    - AI: Juan, Anita and Jim.
    - Web: Juan.
    - Database: Jane.
    - Systems: Bruce.

For scenario 1, since the same number of candidates are hired in all solutions, any one of them is satisfactory. In the case of scenario 2, however, despite the fact that all 42 solutions found satisfy the constraints established, in 6 of them, one candidate fewer is hired, which might indicate a better solution, in terms of preserving funds.

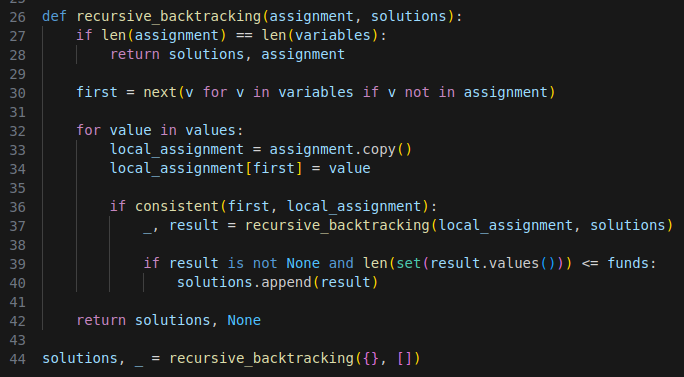
## Question 2

The CSP framework uses a brute-force approach to find a solution to the problem, by testing all possible combinations of variables and their respective values against the constraints. It is only after all possibilities have been exhausted that the method returns the satisfactory solutions found.

As it is often, if not always, the case, brute-force techniques are very expensive as their execution cost tends to grow exponentially. In the cases studied here, scenario 1 has a complexity of 8⁷ (8 values for 7 variables) which leads to 2,097,152 possibilities to be tested, one by one. Scenario 2 is even more complex, with 8 variables and 8 values, therefore 8⁸ = 16,777,216 possibilities to test.

## Question 3

To solve the above scenarios algorithmically, the recursive backtracking algorithm (Abiy *et al*., 2024) was chosen. The below is the implementation of the algorithm in Python:



### Figure 1: Recursive backtracking algorithm implemented in Python.

The variables, domains, constraints and, most importantly, results when applying the method above are the exact same as the framework ones from Question 1. However, it is worth noting that the former outshines the latter by a considerable margin, when it comes to performance. While the framework takes around 9 seconds and over 1.5 minutes to solve scenarios 1 and 2, respectively, it only takes the algorithm a fraction of a second to find the solution in either scenario.

The aforementioned discrepancy in performance between the two techniques can be attributed to the fact that while the framework employs brute force, as discussed in Question 2, the recursive backtracking algorithm structures the problem as a tree that can be pruned to reduce the search scope. Whenever a node that breaches any one of the constraints is reached, the algorithm backtracks and discards the non-compliant branch of the tree altogether without testing every single possibility in it. This approach effectively reduces the domain of each variable to compliant values only by rejecting non-compliant ones on the spot. Therefore:

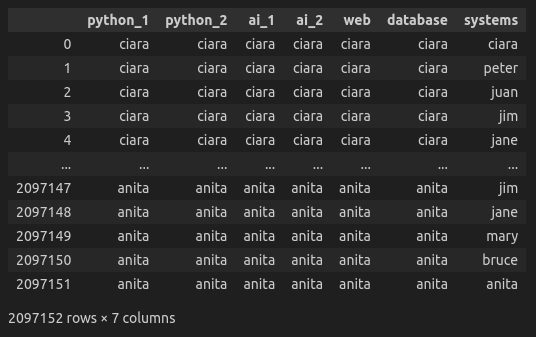
* Scenario 1:
  + Framework: an 8-ary truth-table with 7 columns => 8⁷ = 2,097,152 rows, as seen earlier in Question 2.
  + Algorithm: a 7-layer-deep tree with = (1 x 3 x 4 x 3 x 3 x 1 x 3) = 324 nodes (after pruning), where v is the doman-reduced range of values each variable can assume.
* Scenario 2:
  + Framework: an 8-ary truth-table with 8 columns => 8⁸ = 16,777,216 rows, as seen earlier in Question 2.
  + Algorithm: a 8-layer-deep tree with = (1 x 3 x 1 x 4 x 3 x 1 x 1 x 3) = 108 nodes (after pruning), where v is the doman-reduced range of values each variable can assume.

From the above, it can be seen that the search-scope considered by the algorithm is ~6473 and ~155345 times smaller than their framework counterparts, respectively to scenarios 1 and 2. It is worth noting that the above calculations did not take the funding limitation of each scenario into account, which, being an extra constraint, are likely to reduce the scope even further. Additionally, more sophisticated algorithms/implementations may include some form of heuristics to make the search for a single, albeit not always optimal, solution even faster.

# Tasks for Data Visualisation

## Question 4

The image below shows the dimensions of the “truth-table” for scenario 1, when treated with the CSP framework:



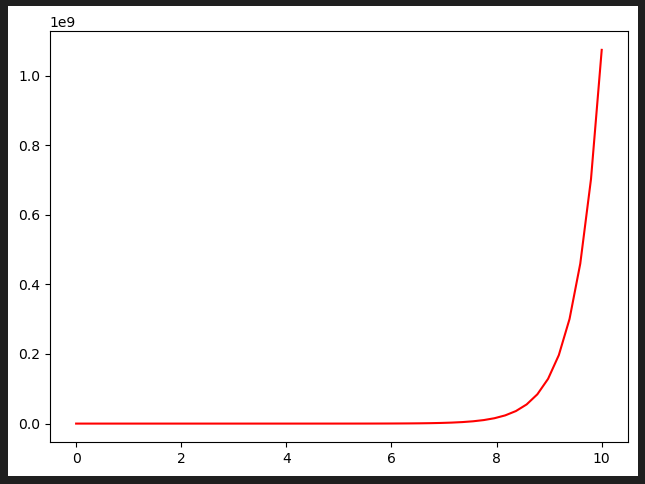
### Figure 2: Scenario 1’s truth table processed by the CSP framework. Note the number of rows.

The next image is analogous to the previous one, but applied to scenario 2:



### Figure 3: Scenario 2’s truth table processed by the CSP framework. Once again, note the number of rows.

The next image shows the exponential growth in size as the number of variables increases:



### Figure 4: As the number of variables increases incrementally, the number of rows in the table increases exponentially.

## Question 5

…

## Question 6

Using Python’s standard GUI framework Tkinter, a frontend application was created that allows users to explore the CSP scenarios, including adding/editing/deleting roles and candidates, as well as selecting the backend (CSP framework or recursive backtracking) and displaying the solutions found, if any.

The images below shows various screenshots of the running application:

| (a) | (b) |
| --- | --- |
| (c) | (d) |
| (e) | (f) |
| (g) | (h) |
| (i) | (j) |

### Figure 5: Screenshots of Tkinter CSP Application. (a) Home screen with empty scenario. (b) Add role option, add candidate is analogous. (c) Load one of the default scenarios. (d) Scenario 1 loaded. (e) Edit/delete candidate option, edit/delete role is analogous. (f) Solve scenario loaded, user is able to choose funds and backend. (g) Progress bar as CSP framework might take over 1.5 minutes. (h) Choose a found solution to inspect. (i) Solutio 3 for scenario 1 displayed. (j) Shown when no solution can be found for a given scenario.

## Question 7

…

# 

# Conclusion

In this assignment, concepts studied in-class when completing the modules were brought together and explored further to deepen understanding.

**\*Personal Note:** unfortunately, I could not find the time I needed to dedicate to this project to get to the results I wanted. I ended up spending too much time with Question 6 and, when I realized, it was too late to do anything meaningful for the other sections.

# References

Abiy, T. *et al*. (2024). *Recursive Backtracking*. Brilliant.org. Available at: <https://brilliant.org/wiki/recursive-backtracking/> (Accessed: 10 January 2024).

Niemeyer, G., Celles S. and Willemsen, F.J. (20023). *python-constraint* (1.4.0) [Computer program]. Available at: <https://python-constraint.github.io/python-constraint/intro.html#download-and-install> (Downloaded: 20 November 2023).