# **CCT College Dublin**

# **Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | Data Visualisation & Communication, Artificial Intelligence |
| **Assessment Title:** | AI\_DV\_Lv8\_ICA\_v5 |
| **Lecturer Name:** | David McQuaid, Sam Weiss |
| **Student Full Name:** | Caroline de Sa Teixeira |
| **Student Number:** | 2020331, 2020317 |
| **Assessment Due Date:** | 05/01/2024 |
| **Date of Submission:** | 05/01/2024 |

### 

### **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. |

# Table of Contents

[Artificial Intelligence 3](#_Toc155353212)

[Data Visualisation & Communication 7](#_Toc155353215)

[Referencing 8](#_Toc155353216)

# Artificial Intelligence

1. Using any CSP (Constraint Satisfaction Problem) framework (using variables, value domains, and constraints), discover if the above problems can be solved and if so, detail who would be in hired**.**

I used the Python constraint model in order to solve a Constraint Satisfaction Problem (CSP) for organizing the logistics of hiring employees for different roles in a company. The breakdown of the code is as follows:

**Variables:**

* *employees*: A list of employee names.
* *roles*: A dictionary assigning each employee to the list of roles they are qualified to fill.
* *scenario1* and *scenario2*: Defines the number of employees needed for each role in two different scenarios.

**Functions:**

* *potentialhires*: Checks if a person can potentially be hired for a specific role based on their skills.
* *constraints*: Adds constraints to the problem for hiring employees for a specific role and count.
* *solve*: Defines and solves the problem using the constraint module, considering the roles and the number of available jobs.

The code contains two scenarios (*scenario1* and *scenario2*) with different hiring requirements, as defined above. It then uses the *solve* function to find solutions for both scenarios.

**Output:**

The script should print the solutions for both scenarios, showing the employees who can be hired based on the constraints. However, I couldn’t make the code work and print the solution.

1. Discuss in detail how using Constraint Satisfaction finds an answer or finds no solution to the problems in Tasks for Artificial Intelligence part 1. How does this differ from standard algorithmic solutions?

Constraint Satisfaction Problems provide a solution by defining a set of variables, their domain values, and constraints that restrict the combinations of values those variables can take. In the context of the given problem, this method is used to find combinations of employees for different roles, satisfying certain hiring constraints.

**Approach:**

* The code begins by defining the *employees* variable and their domains. The *roles* dictionary specifies the roles each employee can fill.
* The functions *potentialhires* and *constraints* are responsible for defining the constraints. It adds variables and constraints for each role based on the qualifications of employees.
* The *solve* function creates a problem instance and adds variables and constraints based on the roles and scenarios.The *solve* function uses the *getSolutions* method.Each solution represents an assignment of employees to roles that satisfies the constraints.
* The script prints the solutions for both scenarios, showing which employees can be hired for each role.

**Differences from Standard Algorithmic Solutions:**

Constraint Satisfaction Problems algorithms, such as backtracking and forward-checking, employ a declarative approach by defining problems with variables, domains, and constraints. Backtracking explores potential solutions through depth-first search, assigning values to variables until constraints are satisfied. If an issue arises, the algorithm backtracks. Forward-checking is a variant that eliminates inconsistent values using constraints.

Constraint Satisfaction Problem algorithms provide a high-level, declarative, flexible, and expressive framework for problem-solving, particularly in the case of combination problems, emphasizing intuitive problem representation and efficient search strategies. In comparison, standard algorithms may require more explicit, step-by-step procedures, which can be less adaptable to changes.

1. These problems be solved using several other algorithm’s we have studied in the module. Choose one of these algorithms and discuss your answer in detail including a proof of your hypothesis in code.

**Dijkstra's Algorithm:**

* The first function takes a graph and a starting node as input and returns the shortest distances from the starting node to the other nodes in the graph.
* It initializes a dictionary called 'distances' to store the shortest distances to each node, setting them to infinity and the starting node to 0.
* It uses a priority queue to explore nodes in order of their distances.
* In the loop, the node with the minimum distance is popped from the priority queue, updating the distances to its neighbours if a shorter path is found.
* The final 'distances' dictionary is returned.
* The *hireEmployees* function uses Dijkstra's algorithm to find the shortest path to employees with the required abilities, selecting the employees based on the available funds.
* It sorts the employees based on their distances from the starting node (Ciara) obtained from Dijkstra's algorithm.
* It iterates through the sorted employees, checking if they have the required abilities and adding them to the list of selected employees depending on the funds.
* The graph represents the employees (nodes) and their abilities (edges with weights).
* Ciara is included in the graph as a starting node with no abilities.
* Two scenarios are defined with specific requirements for the number of employees with certain abilities.
* The funds are also defined for each scenario.
* The code runs the *hireEmployees* function for both scenarios, and the selected employees are printed as output.

Dijkstra's and A\* algorithms can be applied to solve pathfinding problems in graphs, but they might not be the best choice for problems like the one above. The code performs Dijkstra's algorithm by representing the problem as a graph and finding the shortest path to employees, considering the required abilities and funds constraints.

Data Visualisation & Communication

1. Include in your report a section for a theoretical AI “team” you are part of, explaining the visualisation processes and rationalising your visualisation decisions (eg chart choice, colour, layout etc).

My visualization decisions were made based on effective communication through data visualization, with the goal of optimizing role assignments using a Constraint Satisfaction Problem (CSP) approach. The bar chart provides a clear representation of the distribution of roles among team members. Each bar symbolizes a role, with segments indicating individual team members.

Additionally, varying colours are assigned to each team member. The legend on the side facilitates quick reference, displaying team members' names and their colours. The update feature allows users to explore different scenarios by entering the chosen number of team members. Error handling ensures only valid numerical inputs are accepted.

My approach combines dynamic updates, a bar chart, colour mapping, a legend, and error handling, creating a tool for effective communication. However, my Constraint Satisfaction problem didn’t work, so the visualization implemented in the code may not be accurate or aligned with the assignment requirements.

Referencing

1. McQuaid, D. (2023) ‘Constraint Satisfaction in the Real World, Artificial Intelligence, CCT College Dublin, p. (Accessed: 15 December 2023).
2. McQuaid, D. (2023) ‘Constraint Satisfaction with Python', Artificial Intelligence, CCT College Dublin. (Accessed: 15 December 2023).
3. McQuaid, D. (2023) ‘The Australian map-colouring problem’, Artificial Intelligence, CCT College Dublin. (Accessed: 15 December 2023).
4. McQuaid, D. (2023) ‘The eight queens problem’, Artificial Intelligence, CCT College Dublin. (Accessed: 15 December 2023).

#### Constraint-Satisfaction Problems in Python (2022) YouTube. YouTube. Available at: <https://www.youtube.com/watch?v=D1LVbE8nyXs&t=625s> (Accessed: 3 January 2024).

#### Constraint Satisfaction Problems (CSP) in Artificial Intelligence (2023) GeeksforGeeks. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/> (Accessed: 4 January 2024).

1. Popovi&#263;, O. (2023) Constraint Programming with python-constraint, Stack Abuse. Stack Abuse. Available at: <https://stackabuse.com/constraint-programming-with-python-constraint/> (Accessed: 4 January 2024).
2. Weiss, S. (2023) ‘Embedding Visualisations in a GUI - Full, Data Visualisation and Communication, CCT College Dublin. (Accessed: 5 January 2024).