CCT College Dublin

Assessment Cover Page

| Module Title: | AI and DV |
| --- | --- |
| Assessment Title: | Integrated CA 2 |
| Lecturer Name: | Sam Weiss & David Mcquaid |
| Student Full Name | Muhammad Anas Baig |
| Student Number | 2021387 |
| Assessment Due Date | 29/12/24 |
| Submission Date | 29/12/24 |

| 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. |
| --- |

[**Constraint Satisfaction Problem CSP 3**](#_sxj0iglu0u5o)

[CSP Solving Works 3](#_utjsy77xkf9j)

[CSP Solving Works In Our Code 3](#_7a7u6e58dhzb)

[How CSP Differs from Standard Algorithmic Solutions 4](#_ukg5kjpw9icv)

[**Comprehensive Explanation of the CSP Model and Results 4**](#_fdbochl66ktk)

[Defining variables 4](#_x91515ff7bu7)

[Scenario One 5](#_2aikoavzitm5)

[Hiring Justifications for a Solution 5](#_h9z8ve5ttfv4)

[Scenario Two 6](#_te5ftw5sedgm)

[**DFS Discussion 8**](#_qmuaihgeyu8n)

[Scenario 1: Stopping After One Solution 8](#_m2in9m27k2e)

[Scenario 2: Exploring All Solutions 8](#_g6cuici1155c)

[**Data Visualisations 10**](#_wi6z2ocze22p)

[Bipartite Graphs of Roles and Candidates 10](#_h5nxvavp32ou)

[Horizontal Bar Charts for Role-Specific Assignments 12](#_vmdcbgbzwuys)

[Interactive visualization of scenario 13](#_3l9r2abibncy)

[Bar Charts for Role Distribution 14](#_25rwxl9hwc7)

[Altair Interactive Bar Chart 14](#_snb3vheqyd99)

[Grouped Bar Charts for Role Distributions 14](#_jvjvm1tgabl7)

[**Reference: 16**](#_ycd7h51pqfut)

### 

### 

### 

### 

### 

### 

### 

### Constraint Satisfaction Problem CSP

Constraint Satisfaction Problems (CSPs) involve finding values for variables within specific constraints. Common algorithms for solving CSPs include backtracking, forward checking, and constraint propagation techniques like arc consistency. These methods systematically explore possible solutions, ensuring all constraints are satisfied.(GeeksforGeeks, 2023)

##### CSP Solving Works

First we define problem by specifying:

**Variables:** Elements whom to values will be assigned

**Domains:** Type or range of values a variable can accept

**Constraints:**  Rules for assigning value to be followed for assigning values to variables.

Then constraint library uses a combination of techniques to find solution:

**Constraint Propagation:** It is a fundamental concept in constraint satisfaction problems.Basically it start with initial domain of variables then apply constraints to reduce domain of variables, And repeat this process until a state ia reach where no more propagation done(GeeksforGeeks, 2024). For Example: If Ciara is assigned to P1, and the distinct Python programmer constraint is applied, Ciara is removed from the domains of P2 and P3. This reduces the search space.

**Variable and Heuristics:** The order in which variables are chosen and values are assigned can significantly impact performance. The constraint library likely uses heuristics like minimum remaining values or degree heuristic to guide the search.For Example: The lexicographical order constraints added in code p\_order and ai\_order are a form of value ordering, guiding the solver to explore solutions in a certain order to avoid permutations of the same assignment.

**Backtracking** is a depth-first search method that arranges variable values step by step.If at any assignment a constraint returns false the algorithm backtracks to the previous variable and tries a different value. This approach can be inefficient for large problems due to its exhaustive nature.(GeeksforGeeks, 2023)

##### CSP Solving Works In Our Code

With python-constraints:

**Variables**: These are the roles P1, P2, P3, AI1, AI2, AI3, Web, DB, Sys, Sec. P(1,2,3) are Python developers AI(1,2,3) are Artificial Intelligence Engineers. Web is Web developer and DB is Database Sys represents System administrators and Sec do security.

**Domains:** These are the possible values each variable can take the names of the people Peter,Juan,Jim,Jane,Mary,Bruce,Anita and Maria

**Constraints:** These are the rules that restrict the possible assignments of values to variables (e.g., no one can have more than two roles, distinct people for certain roles, Ciara must be P1, Juan must be in a team).

**Searching** :

* The solver will start by assigning P1 = "Ciara" (there’s no choice here).
* Then it tries possible values for P2 in ["Peter", "Jane", "Bruce"].
* For each choice of P2, it tries possible values for P3, ensuring P3 is different from P2 and also respects Ciara < P2 < P3 (the lexicographical constraint).
* It proceeds to assign AI1 and AI2, ensuring AI1 < AI2.
* It assigns Web, DB, and Sys accordingly.
* After each partial assignment, it checks if any constraint is violated. If a partial assignment violates a constraint, the algorithm backtracks and tries another value.if at any point a person is assigned to three roles, the constraint two\_roles fails, and the solver backtracks immediately.
* Invalid options are eliminated early (e.g., two\_roles ensures no individual exceeds two roles).

**Solution or No Solutions:**

If the solver finds a valid solution that is satisfying all the constraints it will return as a solution.On default problem.getsolution returns all the valid assignments but we can stop it as soon as we get a valid solution.  
But if the solver explores all the work space and finds no valid values that satisfy all the constraints it will return as No solution found.

##### How CSP Differs from Standard Algorithmic Solutions

In CSP we define the problem by setting what are the constraints, what are variables and domains but in standard algorithms we specify the exact step by step solution. While CSP focuses on defining and satisfying constraints, standard algorithms often focus on optimising a specific function.CSP algorithms are designed to exploit problem structure which leads to more efficient problem solving. CSP search involves backtracking, propagations on the other hand Standard Algorithm might use direct computation of program.Algorithm is a step-by-step procedure for solving a problem or accomplishing a task.In this kind of problem csp is very flexible cause we can add or remove constraint easily but as compared to standard Algorithm we might need to change whole logic of the algorithm increasing acceptance of variables and things like that.So changing constraints for them may require redesign or developing new procedures.(GeeksforGeeks, n.d.)CSP is capable of finding all possible solution that satisfies the constraints. While normal algorithms generally stop after finding a valid solution. Such as depth-first search, breadth-first search, and A\* search, are general-purpose methods used to explore possible states in a problem space to find a goal state.(Singh, 2024)(Elliott and Mein, 2019)

### Comprehensive Explanation of the CSP Model and Results

This CSP model is designed to solve the optimal allocation of individuals to roles in a hiring scenario. Each role has a specific ability and its constraints. And the solution meets several additional conditions.

##### Defining variables

As we need 3 Python Developers , So P1 means Python Developer position one, P2 and P3 are the rest of positions.,Also for Artificial Specialists we did AI1,AI2,AI3 ,And domains are name of persons that can fill roles ,DB represents database, Web is Web developers and sys is system administrator,Sec is security

##### Scenario One

In this scenario we came to know that Ciara can only hire 4 people due to funds and she knows python.

**Roles and Candidates:**

* P1 : Fixed as "Ciara" due to her expertise and budget constraints As she knows python she might as well be in the team So P1 only domain is Caira .
* P2 and P3 : Candidates include Peter, Jane, and Bruce.
* AI1 and AI2 : Candidates include Peter, Juan, Jim, Anita, and Maria.
* Web : Candidates include Juan, Mary, and Anita.
* DB : Candidates include Jim and Jane.
* Sys: Candidates include Juan, Jim, Mary, and Bruce.

**Constraints:**

**Max Two Roles Per Person:** This makes sure that no individual is hired in more than two roles as the problem requires.As Caira is in the team we didn’t add P1 when calling for constraint because as required she has to hire 4 people other than her so here role is not in checking for this constraint

**Different Candidates for Similar Roles:** This constraints we developed because if a person shouldnt fill same role e.g Peter knows python without this constraint program can fill peter to both python roles according to Max Two Roles constraint which is illogical, So

P2 and P3 must be different individuals.

AI1 and AI2 must also be different individuals.

We didn’t included P1 to be different because of P1 fixed to Caira

**Only Four Distinct People Hired:** Limits the number of hires to four individuals, ensuring budget and team size constraints are respected.

**Lexicographical Order:** Maintains a deterministic order for hiring to simplify evaluation and prevent permutations of the same solution.The lexicographical order constraints ensure that equivalent solutions (permutations) are not counted multiple times.

##### Hiring Justifications for a Solution

For example, consider first solution:

{'P2': 'Jane', 'P3': 'Peter', 'AI1': 'Maria', 'AI2': 'Peter', 'DB': 'Jane', 'Web': 'Mary', 'Sys': 'Mary', 'P1': 'Ciara'}

**Who Gets Hired and Why:**

1. Ciara: Assigned to P1 because she is the only candidate fixed for this role.
2. Jane: Selected for P2 and DB. This satisfies the "max two roles per person" constraint and reflects her eligibility.
3. Peter: Selected for P3 and AI2. Again, this adheres to the "max two roles per person" rule.
4. Mary: Hired for Web and Sys

No one took more than two roles, python developers are different, Ai specialists are different as well.The solution includes only four distinct hires excluding person for role P1 : Ciara, Jane, Peter,Maria and Mary.

##### Scenario Two

Now Ciara and Juan became partners and they need 3 Python Programmers, 3 Al Engineers, 1 Web Designer, 1 Database Admin, and 1 Systems Engineer.

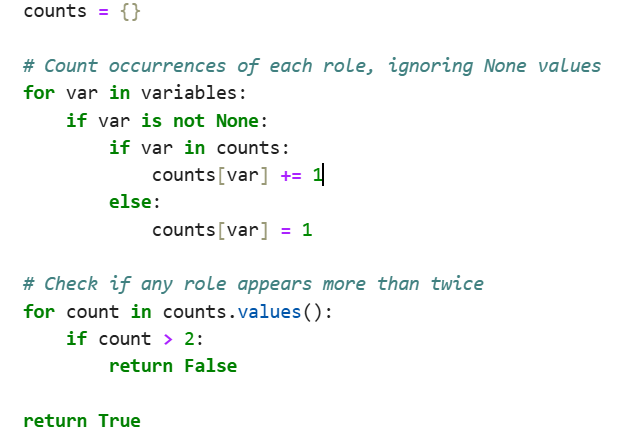
They would also LIKE a Security Employee, but this is not critical

**Roles and Candidates:**

* P1,P2 and P3 : Candidates include Caira,Peter, Jane, and Bruce.In this scenario we haven't fixed Ciara in domains instead we will do by constraint,
* AI1 and AI2 : Candidates include Peter, Juan, Jim, Anita, and Maria.
* Web : Candidates include Juan, Mary, and Anita.
* DB : Candidates include Jim and Jane.
* Sys: Candidates include Juan, Jim, Mary, and Bruce.
* The "Sec" variable includes None as one of the possible values. This means that the "Security" role can either be assigned to one of the available people "Mary", "Maria" or left unfilled (None).

**Constraints:**

**Two\_roles :** Same as scenario one we have two role constraints here as well. The main purpose is to ensure that no one should take more than two roles.



This part has a dictionary so that count for people is managed if some go above 2 is returned false, As well as (None) is handled here which basically means if the sec is none mean no security don't include None as role, similarly in distinct role its handled as well.

**Distinct Role Assignments:**Separate constraints for Python (distinct\_PP) and AI (distinct\_AI) ensure no person is assigned multiple roles within the same category.

**Ciara is Fixed in the Team:** Ciara is mandated to take the P1 position, reason for this is as she is partner she might wanna be in team and she only know python, so she is just gonna fix in python roles, so i fixed here to less the solution space and efficiency

**Juan as a Partner:** The constraint ensures Juan is part of the team by being assigned at least one role. As Ciara Juan is a partner too and he also wants to be in the team but he has more abilities so we can't fix him to a role instead we make sure that he is present in solution. So this constraint makes sure every solution has Juan at least in one role.

**Five Distinct People**: Excluding Ciara and Juan, five distinct individuals are selected for the remaining roles, ensuring diversity.

**Handling Permutations**: Same as scenario one in here we also take care of solutions so that permutations are rejected . Constraints like p\_order and ai\_order ensure lexicographical ordering to avoid duplicate solutions caused by permutations.

**Optional Role:** Security (Sec) is optional and can remain unfilled, adding flexibility to the solution.Including None in the domain of the "Sec" variable ensures that the role can remain unfilled if it doesn't fit the constraints. This adds flexibility to the solution space while still enforcing other rules

**Who Gets Hired and Why:**

'Possible Solution:

'{'AI1': 'Anita', 'AI2': 'Jim', 'AI3': 'Juan', 'P2': 'Jane', 'P3': 'Peter', 'Sec': None, 'Web': 'Anita', 'Sys': 'Mary', 'P1': 'Ciara', 'DB': 'Jane'}

Ciara: Assigned to P1 because she is the only candidate fixed for this role.

Juan is hired because he has to be in team

Jane and Peter are hired in Python roles due to different hires

DB has Jane as well because only 5 person are allowed and jane has his second ability free he was just hired in one role

Anita and Jim are hired as AI roles

Mary is hired in system administration

There is no security hired in this solution

'Possible Solution:

'{'AI1': 'Anita', 'AI2': 'Maria', 'AI3': 'Peter', 'P2': 'Jane', 'P3': 'Peter', 'Sec': 'Mary', 'Web': 'Juan', 'Sys': 'Juan', 'DB': 'Jane', 'P1': 'Ciara'}

But now in this solution we can see still 5 person are hired, two roles constraint is satisfied and we got a Security person which is Mary

### 

### 

### DFS Discussion

##### Scenario 1: Stopping After One Solution

The implementation of this DFS will exit as soon as it will find an answer who satisfies all constraints, it avoids exploring the remaining branches of the search space.

**Features:**

Ensures no duplicate roles for Python (P2, P3) and AI (AI1, AI2).

Enforces lexicographical ordering for Python (P1 < P2 < P3) and AI roles (AI1 < AI2).

Limits each candidate to at most 2 roles.

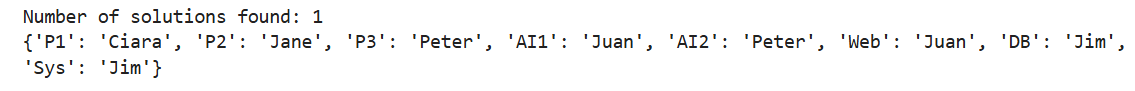
Restricts hires to exactly 4 distinct individuals, excluding Ciara.

The algorithm halts as soon as a valid solution is found, improving efficiency for this scenario where only one solution is needed.

**Analysis:**

Early stopping significantly reduces computational overhead by avoiding exploration of unnecessary branches.I implemented the code so that as soon as it find answer it returns it

**Output:**



**Hypothesis Proof**:The algorithm demonstrates that a solution exists and satisfies the constraints.

##### Scenario 2: Exploring All Solutions

The implementation of this DFS will continue exploring the entire search space even if it will find an answer who satisfies all constraints,it doesn't stop after the first solution and explore all the possible solutions.

**Features:**

Includes a new AI role (AI3), expanding the team to 5 hires plus Ciara and Juan.

Optionally includes a security employee (Sec), with None as a valid choice.

Explores all possible assignments, ensuring that every valid solution is found.

Ensures no duplicate roles for Python and AI.

Enforces lexicographical ordering for Python (P1 < P2 < P3) and AI (AI1 < AI2 < AI3).

Limits each candidate to at most 2 roles.

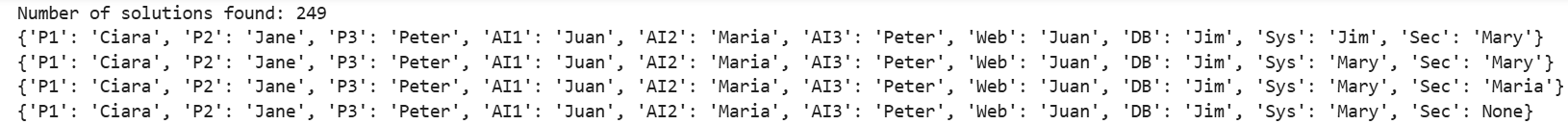
Requires exactly 5 distinct hires, excluding Ciara and Juan.

Ensures both Ciara and Juan are part of the team.

**Analysis:**

This approach is more computationally expensive due to exhaustive exploration but is necessary for comprehensive analysis.Ensures all valid solutions are identified.

**Output:**



**Hypothesis Proof**:The algorithm explores all combinations, ensuring that every valid solution satisfying the constraints is included.

### 

### 

### 

### 

### 

### 

### 

### 

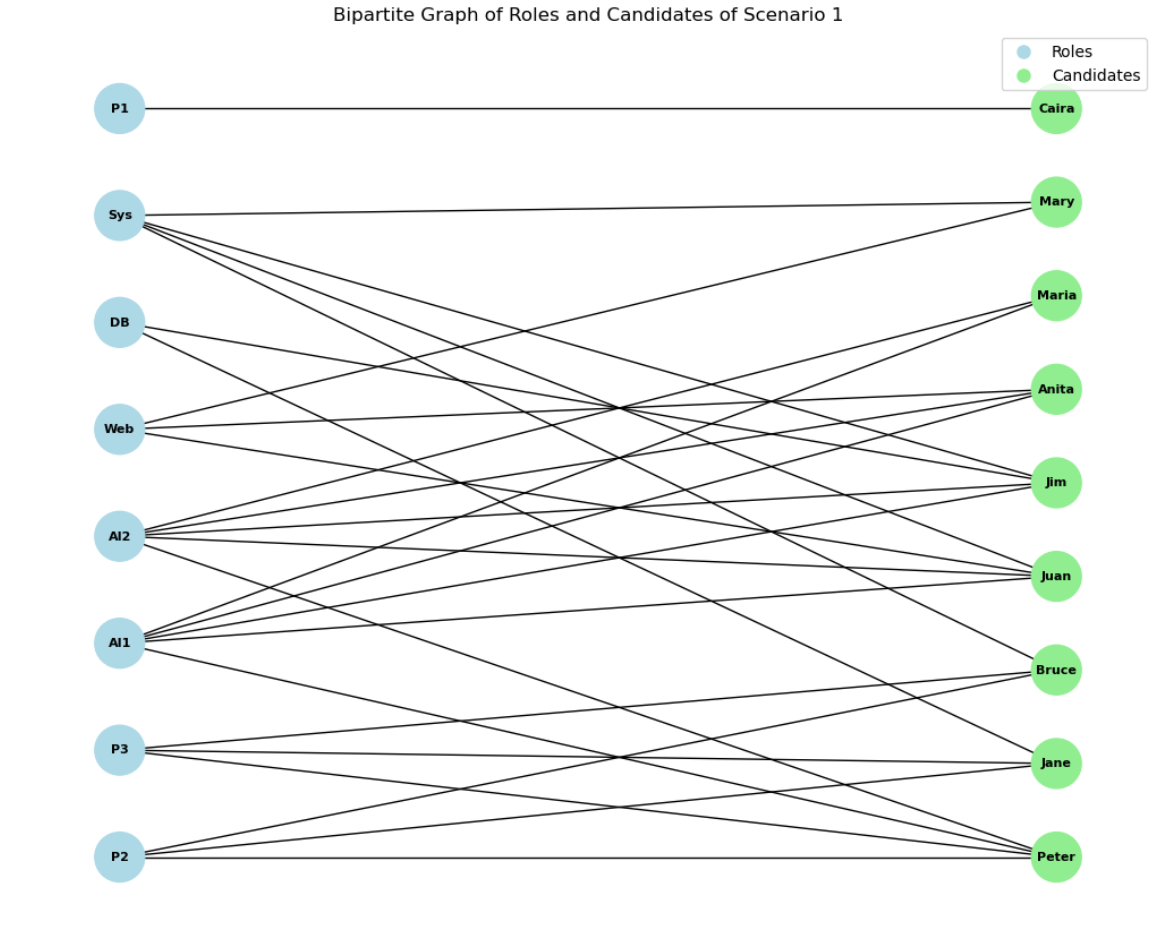
### 

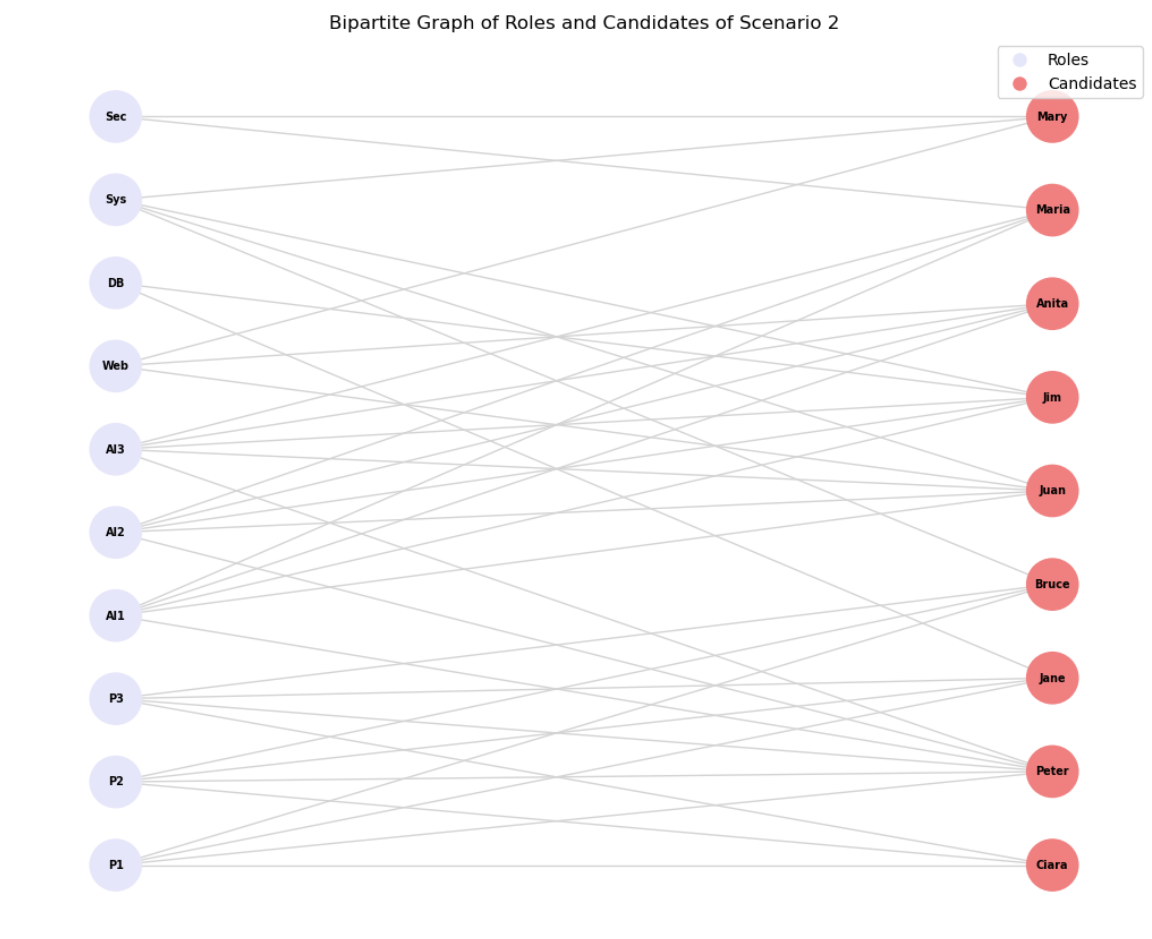
### 

### Data Visualisations

This visualization is for Constraint Satisfaction Problems (CSP) involving roles and candidates, detailing the rationale for each visualization decision.

##### Bipartite Graphs of Roles and Candidates





(Data Science Center, 2022)

We did bipartite graphs of scenario one and two. This will help stakeholders to understand the number of roles and like python positions and AI positions and their eligible candidates easily connected by nodes To represent relationships between two distinct sets of roles and candidates graphically.

This graph was chosen because they effectively depict connections between two groups while maintaining clarity.Roles are represented in light blue, and candidates in light green, ensuring clear differentiation between the two sets.Nodes are aligned in two distinct layers for intuitive visualization of connections.

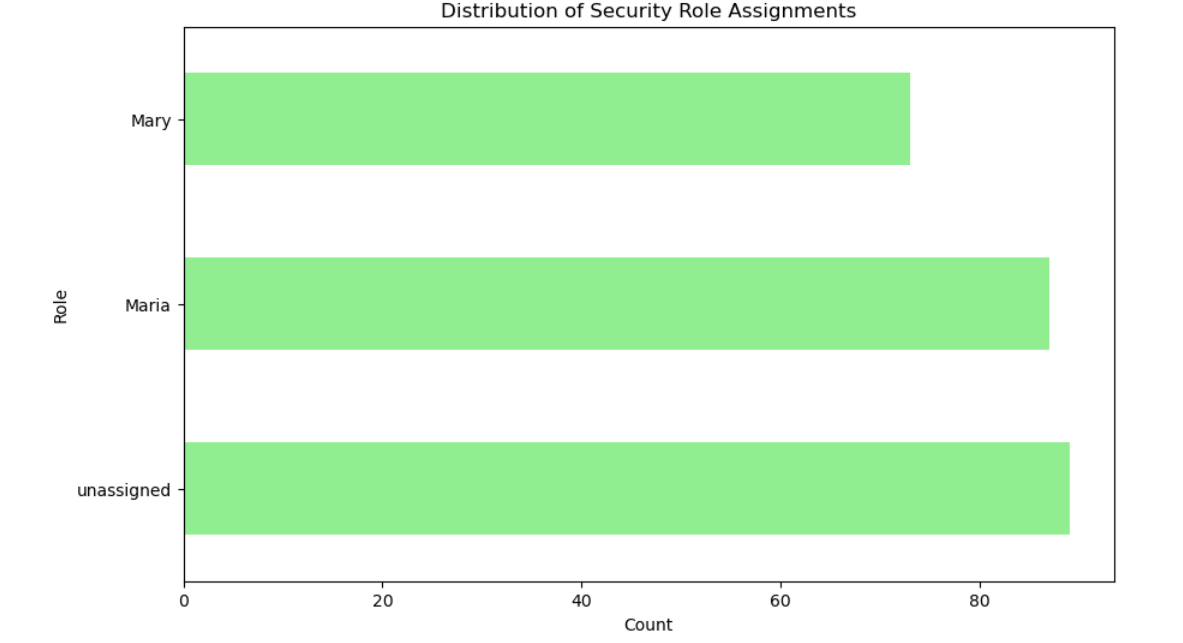
There are solution graphs as well to communicate or develiver who is hired in what role which is same but just solutions here is a random solution graph for that



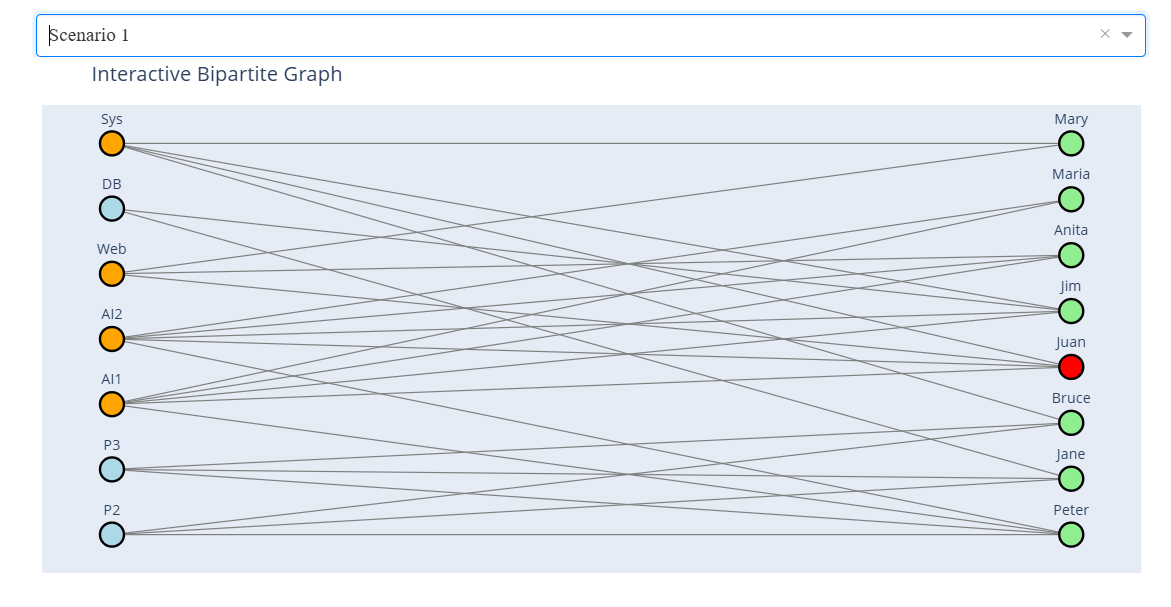
Then we have created an interactive graph which is bipartite as well but it lets the user choose to see full roles and candidates and upon clicking a node it will highlight his connection. Purpose of doing so is to let the user interact with the graph, select a node and see connections distinguished from others.These graphs allow stakeholders to quickly see which candidates are connected to specific roles, making it easier to understand eligibility and assignment options.

##### Horizontal Bar Charts for Role-Specific Assignments

This is a basic graph but it give you a lot more information about scenario two solutions specially of Security, As there was a soft constraint that Security can be None so in graph there are some where Sec is not connected to anything so to see how many solution includes we have this chat unassigned is None value.To stakeholders Bar charts offer a quantitative view of how many candidates are assigned to each role and highlight scenarios where roles remain unfilled the Security role being None.



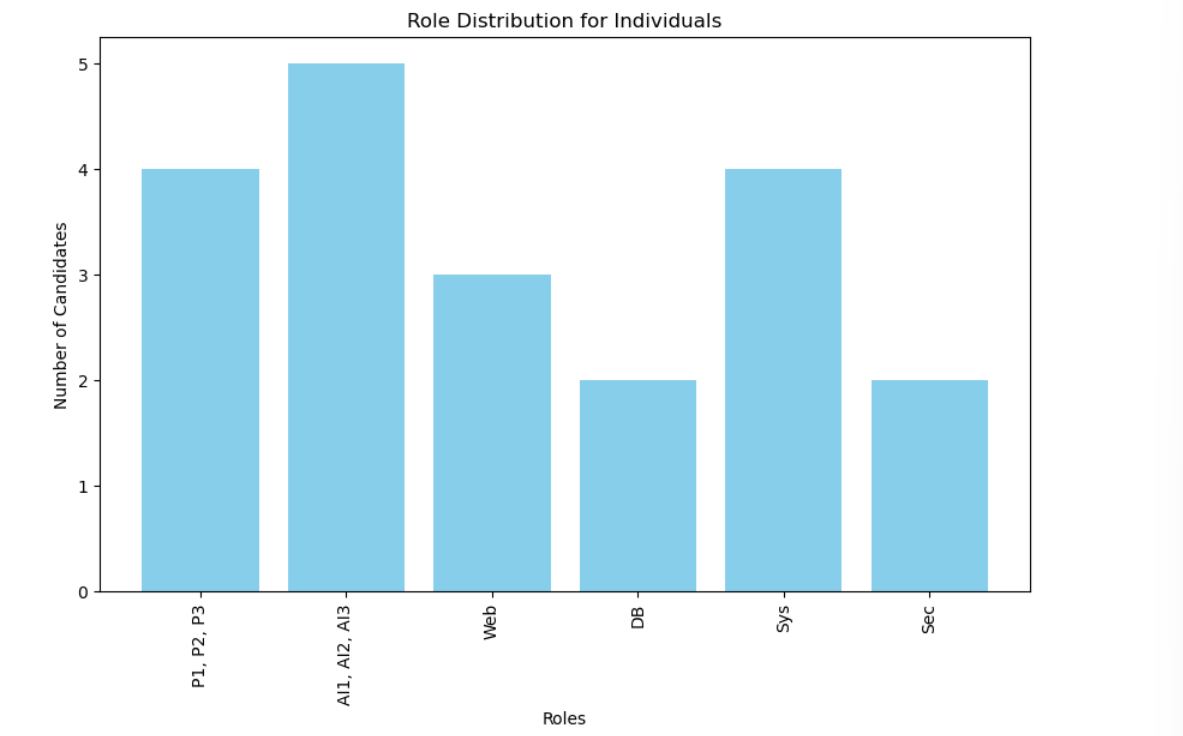
##### Interactive visualization of scenario



Interactivity enhances user experience by allowing the exploration of specific scenarios while maintaining visuals. Interactivity improves understanding by highlighting the relationships between roles and candidates dynamically.Clicked nodes will go red and connection will go orange to clearly distinguish between non selected nodes.For example, clicking on "AI1" highlights the assigned candidate and other connected nodes.

##### 

##### Bar Charts for Role Distribution

To get more deep understanding of how many roles are available per person we have made a bar graph which will explain user overall for what role we have how many options  


We chose a bar chart here because it's easy to understand the availability of candidates for various roles.

Bar charts are effective for categorical comparisons, and the design emphasizes clarity and accessibility.

##### Altair Interactive Bar Chart

This interactive bar chart is actually the reverse of the above bar chart; it gives important information about candidates according to roles. Interactivity in it will clear information about candidates when you hover over it .Conditional formatting was applied to highlight selected candidates dynamically. Tooltips and a well-organized legend were included for clarity and enhanced usability.The interactivity and conditional styling allowed users to explore specific data points.

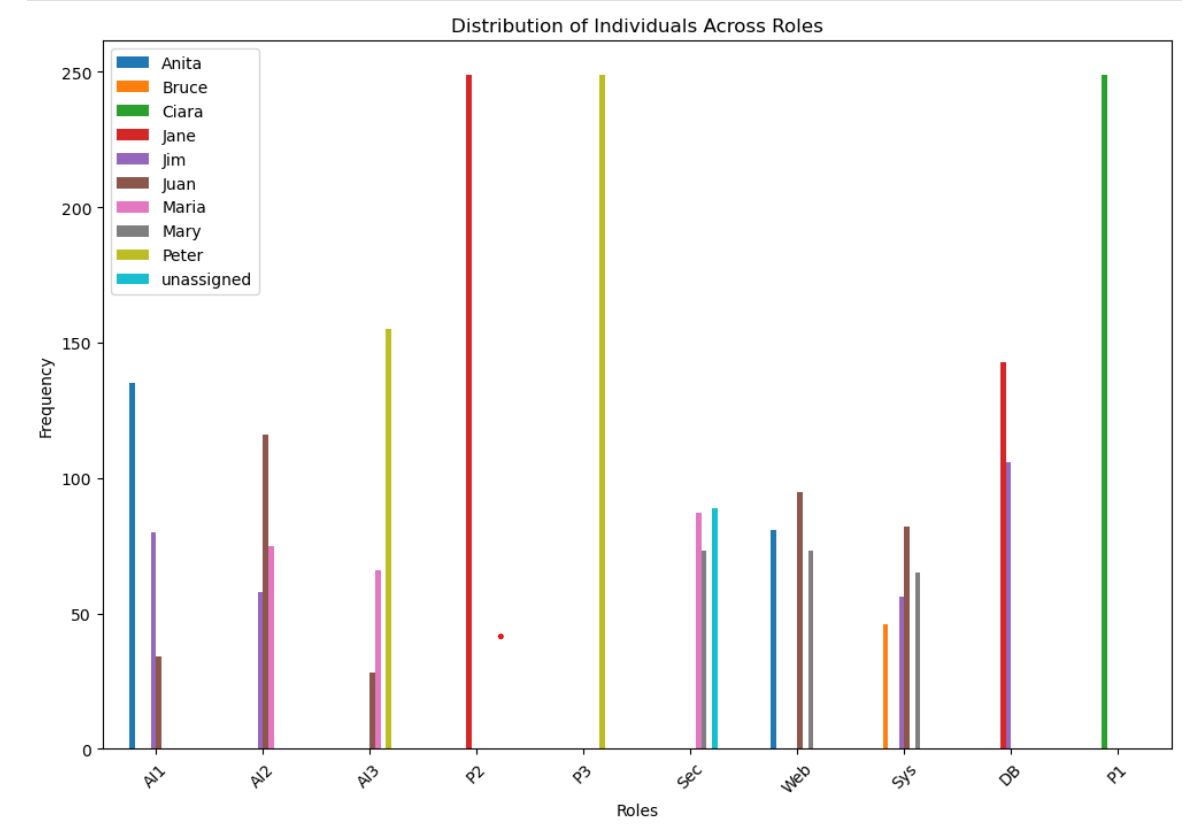
##### Grouped Bar Charts for Role Distributions

This grouped bar chart is based on the solution we got after solving scenario 2. As it talks about solutions it does reflect the CSP how it worked and gave jobs to candidates.

X-Axis Represents the all roles available. And Y-Axis indicates the count or frequency of assignments for each role. Bars of different colours correspond to an individual and their frequency of assignment to a particular role.

So this will help stakeholders to get an idea of Optimized team efficiency, team performance, workload distribution, and role allocation.

Legend will make clear what color bar represents chat candidate



Interactivity in the same graph will let the user zoom in to the bar so if there is minimal difference it can be noticed by zooming in. Also hovering over will give you information of which bar belongs to what candidate and role and also how many times this candidate was selected in these roles in all possible solutions labelled as count.



Stakeholders can evaluate the frequency and distribution of assignments, helping them understand which candidates are frequently preferred for specific roles.

### Reference:

GeeksforGeeks (2024). *Constraint Propagation in AI*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/constraint-propagation-in-ai/.

GeeksforGeeks. (2023). *Constraint Satisfaction Problems (CSP) in Artificial Intelligence*. [online] Available at: <https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/>.

Data Science Center (2022). *Lect 07: Bipartite Graphs with Networkx ||Bipartite Graphs using Python*. [online] YouTube. Available at: https://www.youtube.com/watch?v=FNn8aSi8Dws [Accessed 28 Dec. 2024].

Singh, A. (2024). *Constraint Satisfaction Problems (CSP) in Artificial Intelligence*. [online] Applied AI Blog. Available at: https://www.appliedaicourse.com/blog/constraint-satisfaction-problems-csp-in-artificial-intelligence [Accessed 27 Dec. 2024].

GeeksforGeeks. (n.d.). *Algorithms*. [online] Available at: <https://www.geeksforgeeks.org/fundamentals-of-algorithms>.

Elliott and Mein, C. (2019). *Constraint Satisfaction Problems*. [online] Slideplayer.com. Available at: https://slideplayer.com/slide/15261619/ [Accessed 27 Dec. 2024].