A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name:* | Kevin Scully |
| *Student Number* | 24275 |
| *Module Title* | AI Concepts to Implementation |
| *Assessment Title* | CA2 |
| *Assessment Due Date* | 5th January 2025 |
| *Date of Submission* | 24th December 2025 |
|  |  |

**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.

Contents

[Question 1 3](#_Toc185948657)

[Question 2 3](#_Toc185948658)

[A) Constraint Satisfaction Problem (CSP) Representations 3](#_Toc185948659)

[B) Number of Possible Solutions 4](#_Toc185948660)

[C) Formulation of CSP 4](#_Toc185948661)

[D) Python Code Results 5](#_Toc185948662)

[Question 3 6](#_Toc185948663)

[References 6](#_Toc185948664)

# Question 1

# Question 2

## Constraint Satisfaction Problem (CSP) Representations

A Constraint Satisfaction Problem (CSP) involves assigning values to variables such that all constraints related to these variables are satisfied. This approach is particularly effective for problems like scheduling and timetabling, where tasks must be allocated to resources under certain constraints. (Brailsford, 1999) In a typical CSP, we define three key components: variables, domains, and constraints. (Ghedira, 2013)

For the scheduling problem of assigning doctors to services in a medical clinic, we can identify the following:

* **Variables**: These are the assignments for each doctor on each day, representing the specific task or service a doctor will perform (e.g., routine check-ups, blood tests, surgeries) at a given time.
* **Domains**: The domains for these variables represent the possible values that can be assigned. In this case, the domain might include available doctors and available time slots for the services they perform.
* **Constraints**: These are the rules that must be adhered to in order to ensure a feasible solution. Some of the key constraints in this problem are:
  + Only one doctor will be available for the routine check-up on any given day.
  + Only one doctor can work on both Wednesday and Thursday consecutively.
  + The total hours worked by each doctor across the period must be equal.

The way in which we represent these variables, assign their domains, and express the constraints mathematically or logically significantly affects the complexity of the problem and the efficiency of the solution. The problem could be represented using different types of variables: binary, integer, or categorical. Each representation has its own strengths and trade-offs depending on the requirements of the problem. ()

In a binary variable, each variable, domain and constraint are inputted with a true or false value. For example, if doctor 1 works on Monday, then this is true, otherwise false. This is the simplest way to represent the problem. The benefit of this representation is it is very efficient and computationally inexpensive. As the problem is relatively simple, this representation allows the solver to explore the search space quickly. The drawback of this representation is that it lacks flexibility. If detailed is required on specific hours or services that the doctors can perform, this representation doesn't allow you to model these nuances without adding complexity. ()

The second possible representation is to use integer variables. This is where each variable represents a number. This representation provides more detail. It calculates the number of hours worked by each doctor, which allows for a more direct way to balance the workload based on hours rather than just the number of workdays. This representation is more complex than the binary representation as there are more potential assignments for each variable. It also has an increased search space, making it more computationally expensive. ()

Finally, a categorical representation could also be applied. In this representation each variable represents the service type a doctor performs on a given day. For example, doctor 1 - Monday - routine check-up. This representation is suitable where there is a service specific scheduling required, for example if one doctor can’t perform a certain task. It has flexibility where it can handle cases where a doctor is specialized in certain services. However, this is the most complex and computationally expensive representation. ()

A combination of these representations is preferred because each individual representation addresses a specific part of the problem, and no single representation alone is sufficient to fully model all the constraints. For example, binary variables alone would not allow for tracking hours worked, and categorical variables alone would not provide an easy way to balance the workload. Combining these representations enables us to efficiently capture both the service-specific constraints and the broader scheduling requirements like workload balance.

## Number of Possible Solutions

The number of possible solutions can be calculated by considering the combinations of doctor assignments across the services and days, allowing for restrictions. Following application of the code, there is 280 possible solutions to this problem using CSP for the two-week period.

## Formulation of CSP

A Constraint Satisfaction Problem (CSP) is a mathematical framework used to solve problems by defining variables, domains, and constraints. The objective is to assign values from specified domains to the variables in such a way that all the constraints are satisfied. () In the context of scheduling tasks for doctors at a medical clinic, we can apply the CSP formulation to assign specific services to two doctors, while adhering to various constraints, such as workload balance and availability restrictions.

The precise formulation of the CSP requires well defined variables, domains and constraints. The variables in this problem represent the assignments of doctors to medical services on specific days during specific weeks. These variables capture the task of assigning a doctor to perform a routine check-up, a blood test, or a surgery on a given day. These are as follows:

* Routine\_Checkup\_Monday\_Week(X)
* Routine\_Checkup\_Tuesday\_Week(X)
* Routine\_Checkup\_Wednesday\_Week(X)
* Routine\_Checkup\_Friday\_Week(X)
* Blood\_Test\_Wednesday\_Week(X)
* Surgery\_Thursday\_Week()

The domain for each variable specifies the set of possible values that the variable can take. In this case, the domain will be a set of doctors because the goal is to assign one of the doctors (Doctor 1 or Doctor 2) to each service on a specific day.

The constraints describe the restrictions and rules that must be satisfied in the assignment of doctors to tasks. These are as follows:

* The total hours worked by doctor 1 must be equal to the total hours worked by doctor 2 for the entire period (i.e. the two-week period).
  + H(dr1) = H(dr2) where H is the total number of hours worked.
* Only one doctor can work on both Wednesday and Thursday in a given week, the other doctor cannot work Wednesday and Thursday consecutively
* Only one doctor can be assigned to the routine checkups on a given day.

The precise formulation of the **doctor scheduling problem** as a **constraint satisfaction problem (CSP)** involves clearly defining the **variables**, **domains**, and **constraints** that capture the various aspects of the scheduling task. By ensuring that the model respects the workload balance, availability constraints, and task-specific rules, the CSP framework provides a systematic way to explore the solution space and find a valid, optimal schedule for the doctors. This precise formulation is essential for solving the scheduling problem using CSP solvers, ensuring that all constraints are satisfied while maintaining balance and fairness in workload distribution.

## Python Code Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | Available Hours | Week1 | | Week2 | |
| Dr 1 | Dr 2 | Dr 1 | Dr 2 |
| Monday | Routine Check-up | 7 | 0 | 7 | 0 | 7 |
| Tuesday | Routine Check-up | 7 | 7 | 0 | 7 | 0 |
| Wednesday | Routine Check-up | 7 | 7 | 0 | 7 | 0 |
| Wednesday | Bloods | 4 | 0 | 4 | 4 | 0 |
| Thursday | Surgery | 5 | 0 | 5 | 5 | 0 |
| Friday | Routine Check-up | 7 | 0 | 7 | 0 | 7 |

# Question 3

## Title

Property Titan

## Background

"Property Titan" is a competitive strategy game where multiple players act as real estate moguls in the fictional city of Metropolis. Each player controls an AI-powered agent that buys, sells, and develops properties in order to maximize profits and expand their empire. The value and demand of properties in the game are influenced by factors such as local infrastructure, community popularity, and the location of the properties. Players also have the opportunity to develop infrastructure, such as trains and shopping centres, which enhance the community’s popularity and the attractiveness of locations. However, these investments may also benefit competing players, creating both opportunities and risks.

## Game Rules

1. **Players**: The game is played with multiple players, each controlling an AI agent that represents a real estate mogul. The agents act independently but are directed by the strategic choices made by the players.
2. **Objectives**: The goal of the game is to accumulate the most wealth by acquiring, developing, and selling properties. Players must balance their investments in properties and infrastructure, strategically improving areas to increase the value of their empire while considering how their actions affect the broader market and other players.
3. **Elimination**: Players that can’t afford to repay their debts following investments will enter bankruptcy. This will result in elimination from the game.
4. **Actions**: This is a continuous game. Each player-controlled agent can perform several actions:
   1. **Buy Property:** The agent can bid on properties in different areas of Metropolis based on location, potential value, and market demand. They must out-bid all others in order to secure the property.
   2. **Sell Property:** The agent sells properties to generate capital for reinvestment. They must sell to the highest bidder.
   3. **Develop Property:** The agent upgrades properties to increase their value and attract wealthier buyers.
   4. **Invest in Infrastructure:** The agent can build infrastructure such as trains, shopping centres, and recreational parks. These developments increase the popularity and price of properties in that location. Be careful not to benefit other players who own properties in those areas.
   5. **Evaluate Market:** An AI powered agent can provide reports (for a fee) to analyse market trends and identify the best locations to buy in, and the best time to sell. The AI agent predicts future property values and potential returns on investment.
5. **Constraints:**
   1. **Resource Constraint:** Players have a limited amount of capital to spend each round. They must make strategic decisions about how to allocate resources between property purchases, development, and infrastructure investments.
   2. **Mutual Benefit of Infrastructure:** Infrastructure investments like trains or shopping centers increase the attractiveness of surrounding areas. While this can increase a player’s property values, it may also benefit competing players who own properties nearby. This creates a challenge in balancing short-term gain with long-term competition.
   3. **Market Demand:** The market demand for properties changes dynamically based on player actions and infrastructure investments. A property’s value is impacted by the surrounding infrastructure, such as transport and shopping centres, and fluctuates over time. This constraint reflects how investments and external factors (such as infrastructure) can affect the real estate market and the value of properties. Players must factor in the shifting market dynamics when making investment decisions.
   4. **Bidding Constraint:** When bidding for a new property, a player has a maximum of two bids.

## Definitions

**Intelligence:** Intelligence is reflected in the player’s capacity to understand and navigate the complex market dynamics, predict future property values, and make well-informed decisions based on available data and evolving circumstances. The decision to buy, sell, or develop a property is based on a combination of market analysis and anticipated returns, demonstrating intelligence in optimizing strategies.

**Artificial Intelligence (AI):** In this game, AI is not just used to control non-player agents but also plays a central role in helping players make informed decisions by providing market analysis and predictions. Each player controls an AI agent that independently buys, sells, and develops properties based on a set of programmed strategies that the player can influence. Additionally, an AI-powered agents offer reports on market trends, helping the players predict future property values and market fluctuations.

**Agent:** In *Property Titan*, each player’s AI-powered agent represents a real estate mogul and acts independently within the constraints set by the player. Each AI agent interacts with the game environment (the city of Metropolis) to buy and sell properties, invest in infrastructure, and respond to changes in the market. The agent perceives market conditions, evaluates available properties, and chooses actions to maximize wealth.

**Rationality:** In the context of *Property Titan*, rationality is the principle that drives the agent’s decision-making, with the goal of maximizing profits and expanding the player's property empire. The AI agent is designed to make rational decisions about how to allocate limited resources. The player must carefully allocate capital to different actions, balancing between purchasing new properties, developing existing ones, and investing in infrastructure. The rational behaviour of agents ensures it aims to increase its wealth.

**Logical Reasoning:** In the game, players must use logical reasoning to evaluate market conditions, decide when to buy or sell properties, and understand the consequences of infrastructure development. The players need to reason through the logical implications of their actions. For example, if a player invests in building a shopping center, they must logically assess how this investment will affect property values in the surrounding area, both for themselves and for competing players.

## Conclusion

Property Titan integrates core AI and game theory principles by incorporating intelligent agents, rational decision-making, and logical reasoning. The game’s constraints enforce realistic and strategic gameplay, requiring players to balance investments, assess risks, and adapt to changing market conditions. The presence of artificial intelligence in managing agents’ decisions adds depth to the gameplay and mimics real-world property market dynamics. The result is an engaging, competitive, and strategic experience that emphasizes both player skill and AI-powered decision-making.

# References

**There are no sources in the current document.**