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* | 27th December 2025 |
| *Word Count* | 2,438 |

**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 (561 words) 3](#_Toc186227718)

[A) Min-max values of the root node 3](#_Toc186227719)

[B) & C) Alpha-beta method 3](#_Toc186227720)

[Question 2 (917 words) 6](#_Toc186227721)

[A) Constraint Satisfaction Problem (CSP) Representations 6](#_Toc186227722)

[B) Number of Possible Solutions 7](#_Toc186227723)

[C) Formulation of CSP 7](#_Toc186227724)

[D) Python Code Results 8](#_Toc186227725)

[Question 3 (960 words) 9](#_Toc186227726)

[Title 9](#_Toc186227727)

[Background 9](#_Toc186227728)

[Game Rules 9](#_Toc186227729)

[Definitions 10](#_Toc186227730)

[Game-flow 11](#_Toc186227731)

[Conclusion 11](#_Toc186227732)

[References 12](#_Toc186227733)

# Question 1 (561 words)

## Min-max values of the root node

The min-max evaluates a game tree by assigning scores to possible game states, then works back to the root node to find the best move for both players. (Rivest, 1987) As can be seen below, after applying the min-max on the provided diagram the root node value is 5.

A diagram of a tree

Description automatically generated

## B) & C) Alpha-beta method

Using the Alpha-Beta pruning rules, we identify nodes that don't need to be explored because of the current Alpha and Beta values. These nodes are "pruned." (Fuller, et al., 1973) The following steps were taken in performing the alpha-beta method:

**Step 1: (Alpha = -∞, Beta = ∞)**

Start with the root node, we work down the left branch first.

**Step 2: Left child = 5 (Alpha = -∞, Beta = ∞)**

This is a leaf node with the value of 5. The Alpha value will be updated to 5.

**Step 3: Right child: (Alpha = 5, Beta = ∞)**

Moving to the right child which is blank. No change to the Alpha or Beta values.

A diagram of a network

Description automatically generated

**Step 4. Left child: Max (Alpha = 5, Beta = ∞)**

Moving to the left child and start of left subtree, the node is blank. No change to the Alpha or Beta values.

**Step 5. Left child = 1 (Alpha = 5, Beta = ∞)**

Since 1 is less than the current Alpha = 5, Alpha remains 5.

**Step 6. Right child (Alpha = 5, Beta = ∞)**

Moving to the right child which is blank. No change to the Alpha or Beta values.

**Step 7. Left child = 4 (Alpha = 5, Beta = ∞)**

Since 4 is greater than the current Beta, we update Beta = 4 for this min node.

**Step 8: Right child = 2 (Alpha = 5, Beta = 4)**

Since 2 is smaller than 4, Beta is updated to 2. The min value will be assigned to the blank node as below:

A diagram of a triangle

Description automatically generated

**Step 9. Move back up (Alpha = 5, Beta = 2)**

The node will choose the maximum value between 1 and 2, so Max = 2. No change to the Alpha or Beta values.

A diagram of a tree

Description automatically generated

**Step 10. Move back up and down the right tree (Alpha = 5, Beta = 2)**

Moving up the tree and back down the right sub tree on the left. The left child of the right sub tree (on the min line) is 5. This remains unchanged.

**Step 11. Bottom left child of right sub-tree = 4 (Alpha = 5, Beta = 2)**

Since 4 is greater than the current Beta, this branch is pruned, as the min node will not choose a value greater than 2.

**Step 12. Bottom right child of right sub-tree = 3 (Alpha = 5, Beta = 2)**

Since 3 is greater than the current Beta, this branch is pruned.

**Step 13. Moving back up (Alpha = 5, Beta = 2)**

The max node will choose the maximum value between **5** and pruned branch. Therefore, the value is 5.

**Step 14. Moving back up (Alpha = 5, Beta = 2)**

The min node will choose the smaller value between 2 and 5, therefore 2.

**Step 15. Root node (Alpha = 5, Beta = 2)**

Finally, the root node will choose the maximum value between 5 and 2, therefore the root node is 5.

A diagram of a tree

Description automatically generated

The above shows the values that were not explored due to pruning and the alpha and beta values for the remaining nodes.

# Question 2 (917 words)

## Constraint Satisfaction Problem (CSP) Representations

A CSP involves assigning values to variables so 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, there are three key components, variables, domains, and constraints. (Ghedira, 2013)

For the scheduling problem of assigning doctors to services in a medical clinic, the following variables, domains, and constraints apply:

**Variables**:

* The Medical services
* The day
* The week
* The start and finish times

**Domains**:

* The possible values that can be assigned to the variables.

**Constraints**:

* 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 significantly affects the complexity of the problem and the efficiency of the solution. The problem could be represented differently depending on whether binary, integer, or categorical variable values are used. (Hooker, 2011) Each representation has its own strengths and trade-offs depending on the requirements of the problem.

Binary values input two options for each variable, domain and constraint. 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 detail 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. (Bacchus & Van Beek, 1998)

The second possible representation is to use integer values. This is where each variable represents a number. This representation provides more detail and flexibility. 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. (Tsang, 2014)

Finally, a categorical value could also be applied. In this representation each variable represents the service type a doctor performs on a given day. This representation is suitable for a category of task that the doctor performs. 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. (Rossi, et al., 2006)

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 would not allow for tracking hours worked, and categorical variables would not provide an easy way to balance the workload. Combining these representations will 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. (Mittal & Falkenhainer, 1990) In the context of scheduling tasks for doctors at a medical clinic, the CSP formulation is applied 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.

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 (960 words)

## 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 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 can 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 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. 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 centre, they must logically assess how this investment will affect property values in the surrounding area, both for themselves and for competing players.

## Game-flow

A diagram of a business

Description automatically generated

## Conclusion

Property Titan integrates core AI and game theory principles by incorporating intelligent agents, rational decision-making, and logical reasoning. (Wooldridge & Jennings, 1995) 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

Bacchus, F. & Van Beek, P., 1998. On the conversion between non-binary and binary constraint satisfaction problems.. *AAAI/IAAI,* pp. 310-318.

Brailsford, S. P. C. S. B., 1999. Constraint satisfaction problems: Algorithms and applications. *European journal of operational research,* 119(3), pp. 557-581.

Fuller, S., Gaschnig, J. & Gillogly, J., 1973. *Analysis of the alpha-beta pruning algorithm.* Carnegie-Mellon University: Department of Computer Science.

Ghedira, K., 2013. *Constraint satisfaction problems: csp formalisms and techniques.* s.l.:John Wiley & Sons.

Hooker, J., 2011. *Logic-based methods for optimization: combining optimization and constraint satisfaction..* s.l.:John Wiley & Sons..

Mittal, S. & Falkenhainer, B., 1990. Dynamic constraint satisfaction. *In Proceedings eighth national conference on artificial intelligence,* pp. 25-32.

Rivest, R., 1987. Game tree searching by min/max approximation. *Artificial Intelligence,* 34(1), pp. 77-96.

Rossi, F., Walsh, T. & Van Beek, P., 2006. *Handbook of Constraint Programming..* Netherlands: Elsevier Science.

Tsang,  ., 2014. *Foundations of Constraint Satisfaction: The Classic Text..* Germany: Books on Demand.

Wooldridge, M. & Jennings, N., 1995. Intelligent agents: Theory and practice. *The knowledge engineering review,* 10(2), pp. 115-152.