UNIVERSITY OF GLOUCESTERSHIRE

at Cheltenham and Gloucester

Artificial Intelligence (CT5055)

BSc Computer Science

Table of Contents

Introduction

The application of reliable search algorithms, machine learning models and appropriate frameworks are exclusively addressed and discussed. These components are typically integrated within a semantic network which represents artificial intelligence that tackles both simple and complex challenges. Two theoretical scenarios, each assigned with distinct challenges, are examined and resolved effectively using the previously mentioned topics to deliver efficient outcomes. Artificial intelligence is a significant domain within computer science, having conjured profound transformations in society, shown by the widespread usage of chatbots like ChatGPT, autonomous vehicles and in-built Internet of Things household devices (Russell and Norvig, 2016). By applying these components in the real-world scenarios given, they can reveal substantial implications of their real-world context and get achieve useful efficient results.

These are the two practical scenarios. This includes the configuration of an intelligent traffic management system for smart cities, utilizing a search algorithm, specifically the Hill Climbing algorithm to identify and return the cost-effective optimal path as a result. Swarm intelligence methodologies are implemented to reduce cost by minimizing the relevant cost factors and improving pathway efficiency for vehicles. The second scenario includes forecasting energy consumption of smart buildings using machine learning. Neural networks and the Markov model combined with additional complex mathematical operations are applied for time-series predictions and energy-consumption modelling. A brief description of each scenario is highlighted with required topics followed with a design, a practical solution and a justified conclusion.

Intelligent Traffic Management for Smart Cities

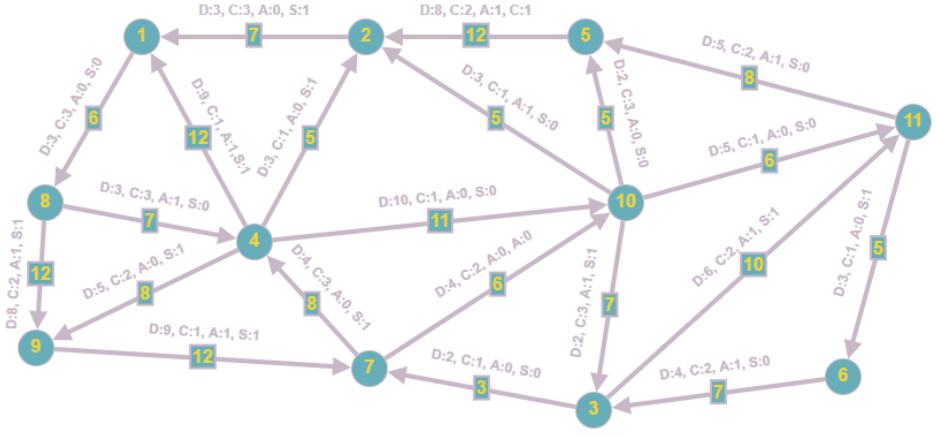

Figure 1 – Graph representation of intersections (nodes) and roads (edges)

Figure 1 reveals a directed, weighted graph that depicts the fundamental road network, where intersections are represented as nodes and the roads connecting these intersections are

represented as edges. Each weight assigned an edge nodes equate to the cost, determined by the combination of the Euclidean distance between nodes, calculated from the formula: $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$ but in this context, the distance is simple along with the traffic congestion, accident probability and road safety. By implementing a search algorithm, an optimal route can be identified for particles to navigate the city efficiently. The optimal route is defined as the pathway that incurs the minimal weight required, or cost in this scenario. For instance, when determining the pathway from Node 1 to Node 10 using Dijkstra's Algorithm - a notable algorithm for identifying the shortest pathway from the chosen source node to the desired target node in a weighted graph - the algorithm performs by producing a queue that selects the next node with the smallest weight cost relative to its neighbor, and a set which stores "visited" nodes. Once a node is visited, it is removed from the queue and added to the set, with the total weight updated. The result was a pathway of node 1 -> 8 -> 4 -> 10, having a total weight count of 24 compared to other weight-consuming pathways such as 1 -> 8 -> 9 -> 7 -> 10 with a total weight count of 36 (6+12+12+6) (Javaid, 2014).

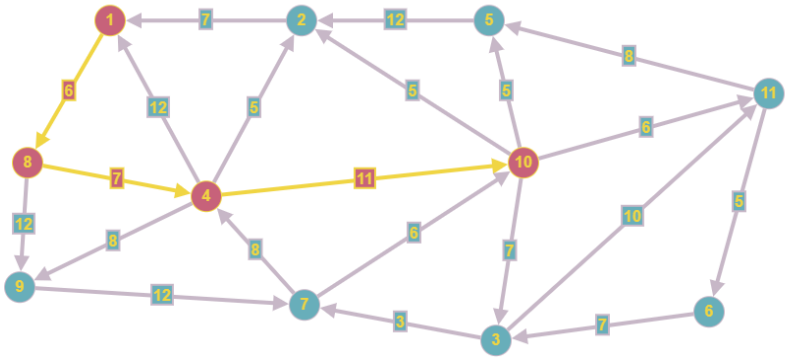

Figure 2 – Applying the Dijkstra's Algorithm on the graph

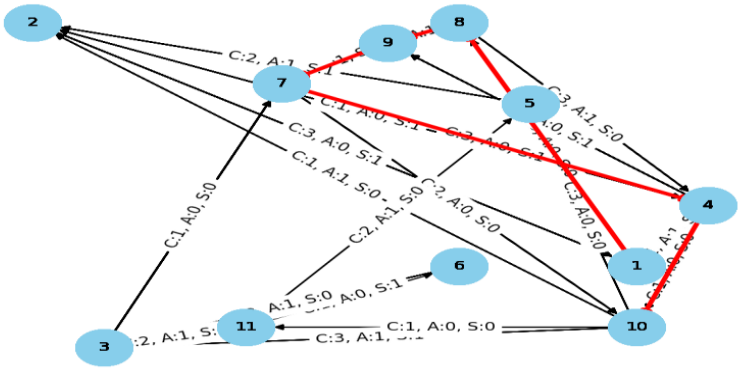

Figure 3 – Visualization of road network using Hill Climbing Algorithm

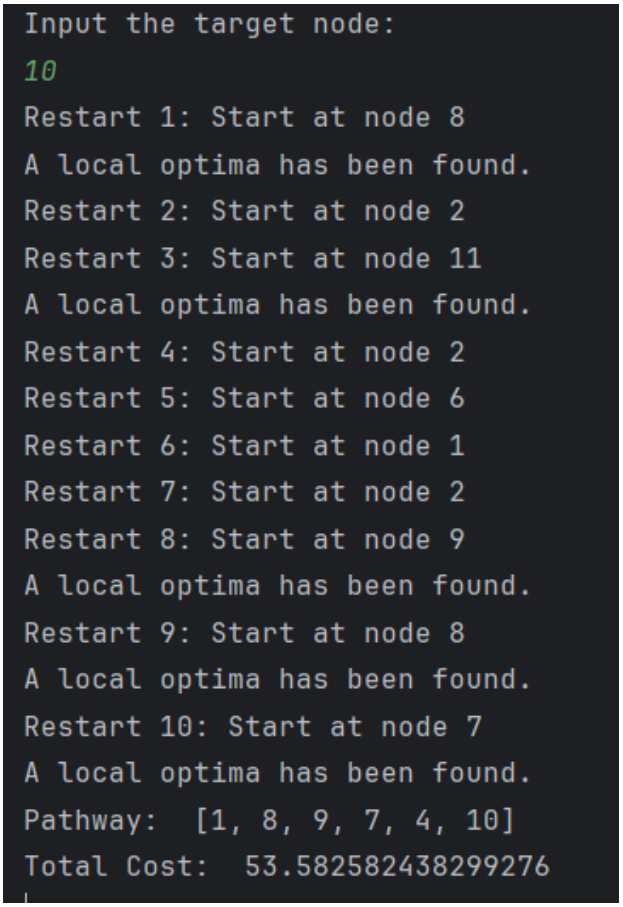

Figure 4 – Command line interface after implementing random restart. Restart 6 is the most optimal pathway

Part A: Search Algorithm for Path Finding

However, the Hill Climbing Algorithm was utilized using Python, providing simpler implementation than Dijkstra's Algorithm. This heuristic local search algorithm uses defined search space and progressively iterates towards the most optimal path-finding solution from the current state. It typically concludes the search when a local optimum is reached, overlooking the global optimum which is a common drawback of the algorithm (Hernando et al, 2018). Starting from the initial node, it determines the next best node by evaluating whether it improves the current result of the fixed heuristic function compared to its current state, repeating this process until no superior sub-pathways are located, ending in local optima or potential global optima yet rare for this algorithm.

Figure 3 and Figure 4 present the Hill Climbing Algorithm operating on the road network from initial node 1 to target node 10. A local optimum has been reached, identifying a limited pathway of node 1 -> 8 -> 9 -> 7 -> 4 along with the associated total cost. Although node 4 is connected to node 10, the current total cost is optimal than the cost from Node 4 to 10, therefore local optima is reached. To mitigate the issue of local optima, the script was modified to incorporate a random restart

feature, allowing the algorithm to initialize from multiple different random nodes and explore unsearched sections of the search space (road network) while keeping track of previously searched pathways to identify and reveal the overall optimal pathway (GeeksforGeeks, 2017).

Part B: Swarm Intelligence for Traffic Optimization

Ant Colony Optimization provides more natural correlation within the road network, as simulated ant agents represent vehicles and the edges symbolize roads. Using pheromone levels as a measure of preferability, it mimics the concept that an increase of vehicles or ants indicates long-term desirability for a pathway (Maniezzo et al, 2004). On the other hand, Particle Swarm Optimization may require deep decomposition to accurately measure the velocity and position vectors of particles within a road network, as it requires continuous graph-based pathfinding (Clerc, 2010). In this context of traffic optimization, ant agents are initialized at various starting nodes, or vehicles, formulating recommended pathways based on associated costs and pheromone levels that typically increases as other vehicles travel. These pathways are assessed based on previously mentioned cost factors – Euclidean distance, congestion, accidental probability and road safety. Additionally, the agents incorporate the total associated cost with pheromone deposits, being inversely proportional to this total associated cost. Consequently, smaller cost-efficient pathways release higher pheromone levels, thereby attracting more ant agents. A cost-based heuristic function is applied to determine the next node for an ant agent, where the heuristic function is defined as the deposits, along with a selection probability function that multiplies the heuristic function by the pheromone level across the set of neighbouring nodes. A greater value from the selection probability enhances the likelihood of a vehicle travelling that pathway. Additionally, pheromone levels on a pathway are updated (Maniezzo et al, 2004)

To prevent long-term accumulation of pheromones due to continuous deposition, a constant evaporation rate was stamped in place, diminishing pheromone levels on each edge. Undesirable edges exhibited low pheromone presence, as agents tend to avoid cost-consuming pathways, thus showing minimal deposits. Termination conditions included a fixed number of simulated attempts (iterations) and the average total cost failing to improve over iterations. To conclude, Ant Colony Optimization proved to be more effective for cost-awareness pathfinding, given that the road-network graph possesses discrete variables such as congestion in contrast to continuous variables like road speed, where Particle Swarm Optimization would excel with maximum performance.

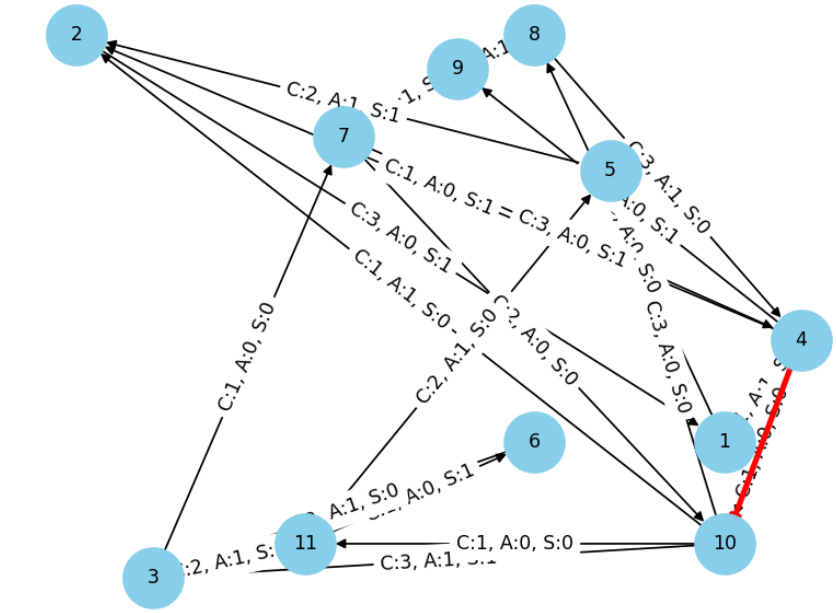

Figure 5 – Visualization of road network using Hill Climbing Algorithm & ACO

```
Enter the target node: 10
Restart 1: Start at node 7
Restart 2: Start at node 3
Restart 3: Start at node 4
Restart 4: Start at node 2
Restart 5: Start at node 12
Restart 6: Start at node 3
Restart 7: Start at node 7
Restart 8: Start at node 11
Restart 9: Start at node 12
Restart 10: Start at node 12

Optimal Pathway: [4, 10]
Total Cost: 7.324555320336759
```

Figure 6 – Command line interface to implement a random restart Hill Climbing algorithm with ACO to reveal overall optimal pathway and optimal cost

Predicting Energy Consumption in Smart Buildings Using Machine Learning

Part C: Recurrent Neural Networks for Time-Series Forecasting

Utilizing the dataset on "Individual Household Electric Power Consumption", we can formulate a neural network model to analyse relevant energy consumption data and continuously train it with historical inputs to predict optimal outcomes overtime. For this dataset, a recurrent neural network will process sequential data, memorizing previous inputs as each neuron communicates required feedback to one another in an incremental fashion. An input is provided, which is passed through a hidden layer to the next timestep after receiving more inputs. The function of the hidden layer evolves as new parameters are introduced and learned by the neural network, therefore the output value is dictated by the updated hidden layer function every timestep (Grossberg, 2013). A basic RNN performs efficiently with short data sequences but often encounters the vanishing gradient issue during prolonged training, as the gradient of changes in hidden states are multiplied repetitively, causing a nearly negligible derivative which fails to influence model training. Long data sequences are ineffective rather variations of RNN, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) are more effective to deal with vanishing gradients. For this scenario, a Gated Recurrent Unit (GRU) is used to predict long-term energy consumption, as it is rapid to train due to fewer gates involved compared to LSTM while achieving similar results (Chung et al, 2014).

The dataset undergoes a process of cleaning and normalization, followed by the conversion of data into sequences for modelling. These sequences are incorporated to predict future sequences, with the model primarily targeting global active power values, reflecting the useful power consumption within a household. Additionally, other factors such as voltage and sub-metering are inputs for predicting future global active power values. During model training, the input layer is succeeded by a single hidden layer that processes the sequences until the final timestep reaches the output layer. The average squared mean difference between predicted global active power values and actual global power values is calculated, aiming to minimize errors between these predicted and actual values as the training function iterates over time using mean squared error. This model predicts global active power values based on provided test data, which presents a limitation, potentially predicting values based on new inputs. A graph was generated to represent the comparison between the predicted values of the model and the actual global active power values, depicted in Figure 7. Initially, a significant disparity between the values is observed, indicating that the model is underperforming, which is standard; but over time, it is expected to align with actual value patterns as the model continues to learn shown in Figure 8. Several hyperparameters impacts were recognised that influenced model performance. A high sequence length resulted in memory issues, leading to segmentation faults due to multiple timesteps while a low sequence length produced slow to no improvement in predicted values. Other parameters included the

training cycles, and the number of neurons present within the hidden layer, where increasing these parameters enhanced model performance but increases the risk of overall model failure.

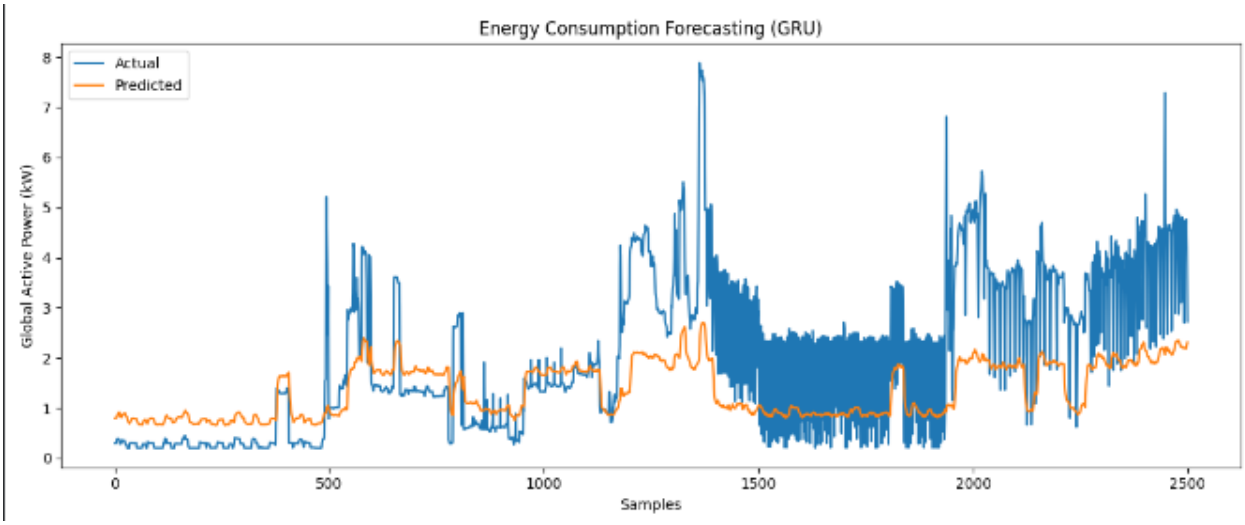

Figure 7 – Visual graph showing the comparison between predicted and actual global active power consumption (Learning Attempt 1)

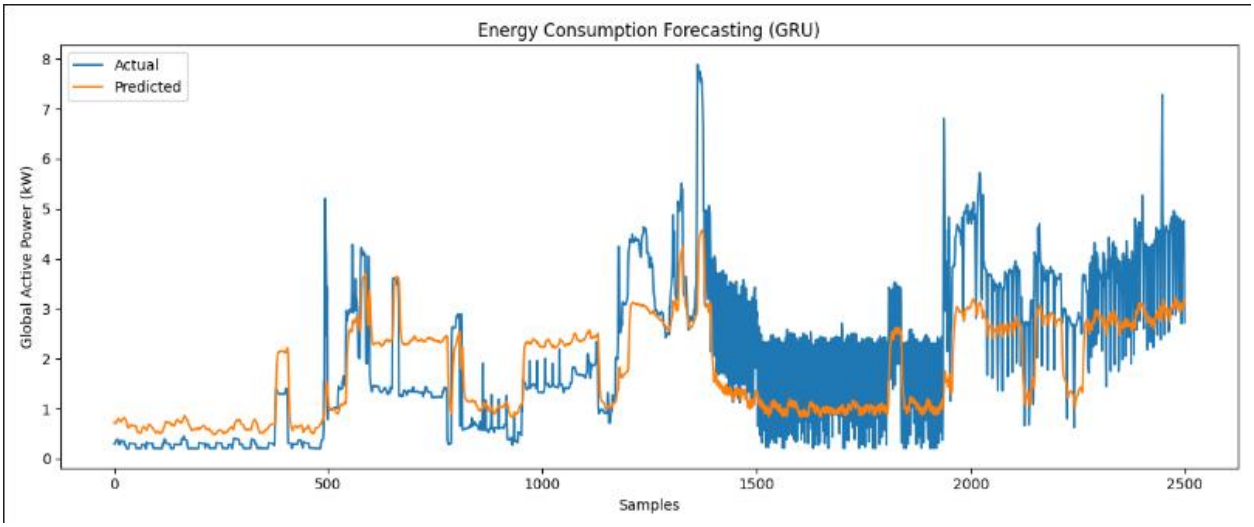

Figure 8 – Visual graph showing the comparison between predicted and actual global active power consumption (Learning Attempt 4)

Part D: Markov Decision Process for Energy Consumption Modelling

A mathematical model implementing the Markov Decision Process technique was developed to predict energy consumption values. In this context, the state for energy consumption would be the specific datetime when certain energy usage levels occur, indicating possible actions to commit maintenance, increase or reduce energy usage based on current state. A transitional probability is

calculated to predict energy usage levels from one datetime to another, denoted as P(d2 | d1, a), where d1 and d2 represent the datetimes (states) and a represents the specific action, used to develop a transitional matrix. Furthermore, a reward function is incorporated to incentivize lower energy consumption, returning a positive reward reflected through a cost increase. Ultimately, a policy is established and improved iteratively until the optimal policy is achieved, enabling the agent to select actions based on the current state (Sutton and Barto, 2018).

Conclusion

To summarize, artificial intelligence incorporates a range of subfields associated with machine learning and algorithms, aimed at extracting valuable insights from unprocessed sequential data to develop sophisticated solutions and frameworks for real-world applications, such as traffic management and predicting energy consumption. As technology advances in discovering more efficient methods for training these virtual entities/agents, these current approaches are useful.

Reference List

- Grossberg, S., 2013. Recurrent neural networks. *Scholarpedia*, *8*(2), p.1888.

- Chung, Junyoung, et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." *ArXiv.org*, 11 Dec. 2014, arxiv.org/abs/1412.3555.

- Clerc, Maurice. *Particle Swarm Optimization*. Wiley-ISTE, 24 Feb. 2010.

- GeeksforGeeks. "Introduction to Hill Climbing | Artificial Intelligence." *GeeksforGeeks*, 12 Dec. 2017, www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/.

- Hernando, Leticia, et al. "Hill-Climbing Algorithm: Let's Go for a Walk before Finding the Optimum." *IEEE Xplore*, 1 July 2018, ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8477836.

- Javaid, Adeel. *Understanding Dijkstra's Algorithm*. 2014.

- Maniezzo, Vittorio, et al. “Ant Colony Optimization.” *New Optimization Techniques in Engineering*, 2004, pp. 101–121, https://doi.org/10.1007/978-3-540-39930-8_5. Accessed 19 Aug. 2021.

- Russell, Stuart J, and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed., Upper Saddle River, Pearson, 2016.

- Sutton, Richard S, and Andrew Barto. *Reinforcement Learning: An Introduction*. Cambridge, Ma ; London, The Mit Press, 2018.