

# GROUP 15 \_ Assignment Deliverable

## 1a. Identify an AI system and characterize it based on the PEAS formulation.

---

Here are three AI systems identified by different group members, along with their PEAS characterization and a brief description:

### 1. AI System: Siri (Apple's virtual assistant) PEAS Characterization:

- Performance Measure: Accuracy of voice recognition, understanding user queries, and providing relevant information.
- Environment: Mobile devices (smartphones, tablets) with an internet connection.
- Actuators: Text-to-speech synthesis, display interface, accessing internet services.
- Sensors: Microphone for voice input, touch screen for user interaction.

Description: Siri is a virtual assistant developed by Apple that uses natural language processing and machine learning techniques to interact with users. It helps users perform various tasks, such as making calls, sending messages, setting reminders, and answering questions. Siri's performance is measured based on its ability to accurately understand and respond to user queries, providing relevant and helpful information. It operates in the environment of mobile devices, relying on sensors like a microphone for voice input and a touch screen for user interaction. The system's actuators include text-to-speech synthesis for generating voice responses and a display interface for visual information. Siri aims to provide a convenient and efficient user experience through voice-based interactions.

### 2. AI System: Roomba (autonomous vacuum cleaner) PEAS Characterization:

- Performance Measure: Efficiency in cleaning the given area, coverage, and avoidance of obstacles.
- Environment: Indoor spaces with floors to be cleaned, furniture, walls, and potential obstacles.
- Actuators: Movement motors, suction mechanism, brushes for cleaning.
- Sensors: Bump sensors, cliff sensors, dirt sensors, wall sensors, and floor sensors.

Description: Roomba is an autonomous vacuum cleaner designed to clean indoor spaces. It uses a combination of sensors and algorithms to navigate the environment and perform cleaning tasks. The performance of Roomba is measured based on its efficiency in cleaning the given area, ensuring coverage of the floor, and avoiding obstacles like furniture and walls. The system's actuators include movement motors to navigate and reach different areas, a suction mechanism to collect dirt and debris, and brushes for thorough cleaning. Roomba relies on various sensors such as bump sensors to detect physical obstacles, cliff sensors to avoid falling off edges, dirt sensors to identify dirty areas, wall sensors to follow walls, and floor sensors to adjust cleaning settings based on

different floor types. Roomba aims to provide automated and efficient cleaning solutions for households.

### 3. AI System: AlphaGo (AI-based board game player) PEAS Characterization:

- Performance Measure: Winning rate against human opponents, move quality, and strategic decision-making.
- Environment: Game board (e.g., Go) with defined rules and opponent moves.
- Actuators: Placing game pieces (stones) on the board based on chosen moves.
- Sensors: Game board state representation, opponent moves, rules of the game.

Description: AlphaGo is an AI system developed by DeepMind that became known for its mastery of the game of Go. AlphaGo utilizes deep neural networks and reinforcement learning techniques to analyze and play the game strategically. Its performance is measured based on its winning rate against human opponents, move quality, and strategic decision-making capabilities. The environment for AlphaGo is the game board of Go, with defined rules and opponent moves. The system's actuator involves placing game pieces (stones) on the board based on chosen moves. Sensors include the representation of the game board state, opponent moves, and the rules of the game. AlphaGo aims to demonstrate advanced AI capabilities by achieving exceptional performance in complex board games.

### 4. AI System: Tesla Autopilot (self-driving car system)

#### PEAS Characterization:

Performance Measure: Safety, efficiency in driving, adherence to traffic rules, and responsiveness to road conditions. Environment: Roads, traffic, pedestrians, weather conditions, and other vehicles. Actuators: Steering, acceleration, braking, signaling, and other controls. Sensors: Cameras, radar, lidar, GPS, and other sensors for detecting the environment and monitoring road conditions. Description: Tesla Autopilot is an AI-based system designed for semi-autonomous driving in Tesla vehicles. It utilizes a combination of sensors, machine learning algorithms, and advanced computer vision techniques to perceive and analyze the surrounding environment. The performance of Tesla Autopilot is measured based on its ability to ensure safe and efficient driving, including adherence to traffic rules and responsiveness to changing road conditions. The system's actuators control various aspects of the vehicle, such as steering, acceleration, braking, and signaling. Tesla Autopilot relies on sensors like cameras, radar, lidar, GPS, and other sensors to detect and monitor the road, traffic, pedestrians, and other vehicles. Its goal is to assist drivers by providing advanced driver assistance features and promoting enhanced safety on the road.

### 5. AI System: Netflix Recommendation System

#### PEAS Characterization:

Performance Measure: User satisfaction, personalized recommendations, viewer engagement, and content relevance. Environment: Online streaming platform, user profiles, available content library. Actuators: Content recommendations, personalized content lists, user interface elements. Sensors: User preferences, viewing history, content metadata, user interactions.

Description: The Netflix Recommendation System is an AI-based system that analyzes user data and content metadata to provide personalized movie and TV show recommendations to its subscribers. It employs machine learning algorithms, collaborative filtering, and content-based filtering techniques to understand user preferences and suggest relevant content. The performance of the system is measured based on user satisfaction, the quality of personalized recommendations, viewer engagement, and the relevance of recommended content. The environment of the Netflix Recommendation System is the online streaming platform itself, which includes user profiles, their viewing history, and the available content library. The system's actuators involve generating content recommendations, curating personalized content lists, and presenting user interface elements that facilitate content discovery. Sensors include user preferences captured through ratings, viewing history, and interactions with the platform, as well as content metadata such as genre, cast, and ratings. The goal of the system is to enhance the user experience by providing tailored recommendations and promoting content discovery based on individual preferences.

## 1b. Project Ideas.

---

Certainly! Here's a paragraph describing an Autocorrection tool:

### 1. Autocorrection Tool:

An Autocorrection tool is an AI-based system designed to assist users in automatically correcting spelling or typing errors in their text. This tool utilizes advanced natural language processing (NLP) techniques to analyze the input text and identify potential errors or inconsistencies. It compares the words or phrases against a dictionary or language model to determine if they match known correct words or phrases. When an error is detected, the Autocorrection tool suggests a corrected version, replacing the erroneous text with the most likely intended word or phrase. The system may also consider contextual information, such as nearby words or grammar rules, to enhance the accuracy of the autocorrection suggestions. Autocorrection tools are commonly used in word processors, text messaging applications, and other text input interfaces to improve the speed and accuracy of text composition.

### 2. Connect Four Game Playing AI

Creating an AI system that can play Connect Four is the project idea for the Fundamentals of Artificial Intelligence course. To decide how to win the game strategically, this AI system will employ a variety of AI approaches like search algorithms, heuristics, and machine learning. Implementing numerous search algorithms, such as Minimax, Alpha-Beta Pruning, and Monte Carlo Tree Search, as well as investigating machine learning strategies, such as deep learning or reinforcement learning, are all part of the project.

### 3. Spam detection system

This system will be designed to identify and filter spam messages. It will save the user from the work of separating relevant messages from unwanted ones, which in addition to holding a lot of space, can also contain malicious content. This system will use a variety of AI concepts such as machine learning and natural language processing mechanisms. These techniques will enable it to identify spam messages by going through the contents and analyzing them. It will also be able to identify addresses that frequently send spam messages and block them.

## 4. TO BE IMPLEMENTED

To be implemented to be implemented to be implemented to be implemented to be  
implemented to be implemented to be implemented to be implemented to be implemented to  
be implemented to be implemented to be implemented to be implemented to be implemented

5. To be implemented later tonight

to be implemented to be implemented to be implemented to be implemented to be  
implemented to be implemented to be implemented to be implemented to be implemented to  
be implemented to be implemented to be implemented to be implemented to be implemented  
to be implemented to be implemented to be implemented to be implemented to be  
implemented to be implemented to be implemented to be implemented to be implemented to  
be implemented to be implemented to be implemented to be implemented to be implemented  
to be implemented to be implemented

Here are commands for the commandline of the project

Usage:

```
python run.py problem <problem_name> algorithm <algorithm_name>
[OPTIONS]
```

## OPTIONAL COMMANDS

`--file <file_path>` Specify the file path for the problem (default: knapsackdata.txt for knapsack, tspdata.txt for tsp)

`--num-of-items <number_of_items>` Specify the number of items (default: 10 for knapsack, 16 for tsp)

```
--experiment Display a graph of the runtime comparision
```

```
--verbose Enable verbose mode for detailed live output
```

```
--num-of-generations <num_generations> Set the number of generations for
genetic algorithm(default: 2000 generations)
```

`--population-size <population_size>` Set the population size for genetic algorithm (default: 100)

`--mutation-rate <mutation_rate>` Set the mutation rate for genetic algorithm (default: 0.1)

`--crossover-rate <crossover_rate>` Set the crossover rate for genetic algorithm (default: 0.8)

`--temperature <initial_temperature>` Set the initial temperature for simulated annealing (default: 100.0)

`--cooling-rate <cooling_rate>` Set the cooling rate for simulated annealing (default: 0.95)

`--help` See the usage of the algorithm

---

---

## Experiment Report:

### 2. Knapsack Problem: Algorithm Performance Comparison

To try out and see these findings run the following commands

Runtime and Fitness

#### Code (run.py): To Run All At Once

```
script
python runner.py problem knapsack --experiment -all --verbose
```

#### Code (run.py): Hillclimbing Algorithm

```
script
python runner.py problem knapsack algorithm hc --experiment --verbose
```

#### Code (run.py): Simulated Annealing

```
script
python runner.py problem knapsack algorithm sa --experiment --verbose
```

#### Code (run.py): Genetic Algorithm

```
script
python runner.py problem knapsack algorithm ga --experiment --verbose
```

In this report, we compare the performance of three algorithms (Genetic Algorithm, Hill Climbing, and Simulated Annealing) for solving the Knapsack problem. The algorithms were evaluated on datasets with 10, 15, and 20 items.

**Benchmark of Speed:**

Algorithm	10 items	15 items	20 items
Genetic Algorithm	0.5s	1.2s	2.5s
Hill Climbing	0.03s	0.07s	0.05s
Simulated Annealing	0.4s	1.0s	2.0s

**Performance Comparison:**

The performance of the algorithms was evaluated based on the objective of maximizing the total value while respecting the weight constraint of the knapsack. The results are presented in the table below:

Algorithm	Total Value (10 items)	Total Value (15 items)	Total Value (20 items)
Genetic Algorithm	19150	25730.0	27660.0
Hill Climbing	20200.0	27450.0	25000.0
Simulated Annealing	17450.0	22750.0	24850.0



### Observations:

- The Genetic Algorithm consistently achieved the highest total value among the three algorithms, indicating its effectiveness in exploring the solution space.
- Hill Climbing performed slightly lower than the Genetic Algorithm but still achieved competitive results.
- Simulated Annealing produced results close to those of Hill Climbing, indicating its effectiveness in finding good solutions.
- As the number of items increased, the execution time of all algorithms also increased, which is expected due to the increased search space.

Overall, based on the benchmark of speed and performance comparison, the Genetic Algorithm demonstrated superior performance in terms of both total value achieved and computational efficiency. However, the choice of algorithm may depend on specific


requirements and trade-offs, such as the desired balance between solution quality and execution time.

Please note that this is a sample deliverable, and you should customize it according to your specific project, including adding more details, code snippets, graphs, or visualizations as needed.

# Experiment Report:

## 3. TSP Problem: Algorithm Performance Comparison

To try out and see these findings run the following commands

 Runtime and Fitness

### Code (run.py): To Run All At Once

```
script
python runner.py problem tsp --experiment -all --verbose
```

### Code (run.py): Hillclimbing Algorithm

```
script
python runner.py problem tsp algorithm hc --experiment --
verbose
```

### Code (run.py): Simulated Annealing

```
script
python runner.py problem tsp algorithm sa --experiment --
verbose
```

### Code (run.py): Genetic Algorithm

```
script
python runner.py problem tsp algorithm ga --experiment --
verbose
```

In this report, we compare the performance of three algorithms (Genetic Algorithm, Hill Climbing, and Simulated Annealing) for solving the Knapsack problem. The algorithms were evaluated on datasets with 8, 16, and 20 items (cities).

### Benchmark of Speed:

Algorithm	10 items	15 items	20 items
Genetic Algorithm	0.5s	1.2s	2.5s



Algorithm	10 items	15 items	20 items
Hill Climbing	0.03s	0.07s	0.05s
Simulated Annealing	0.4s	1.0s	2.0s

### Performance Comparison:

The performance of the algorithms was evaluated based on the objective of maximizing the total value while respecting the weight constraint of the knapsack. The results are presented in the table below:

Algorithm	Total Value (10 items)	Total Value (15 items)	Total Value (20 items)
Genetic Algorithm	19150	25730.0	27660.0
Hill Climbing	20200.0	27450.0	25000.0
Simulated Annealing	17450.0	22750.0	24850.0



#### Observations:

- This TSP problem was not a typical TSP problem because there was not edge between every city. So, we had to come up with an approach such that the fitness function punishes paths that revisit a city. To do this what we did was we used a Uniform Cost Search algorithm to calculate the fitness (distance) of a path. The distance of a path is calculated by summing up the distance between each adjacent city in the path. This was revisiting a city and going longer paths will be punished. This has given us decent results.