**Lab Assignment NO#3**

**Teacher Name: Dr Zunaira Rauff**

**Subject : Artificial Intelligence (AI) Topic:**

**Implementing Multiple Search algorithms we have studied on US Criminals Data Set.**



**Comsats University Islamabad, Attock Campus**

**The US Criminal Analysis Projects  Supervisor:**

1. [**Madam Zunaira Rauf (PHD)**](https://scholar.google.com/citations?user=f2QH4MsAAAAJ&hl=en)

**Students:**

**Name                       Registration Number**

1. [**Hammad Nawaz**](https://pk.linkedin.com/in/hammad-ahmad-khan-693845129) **FA22-BCS-031**

1. [**Muhammad Sadiq**](https://pk.linkedin.com/in/hammad-ahmad-khan-693845129) **SP23-BCS-021**

1. [**Hammad Khan**](https://pk.linkedin.com/in/hammad-ahmad-khan-693845129)  **SP23-BCS-006**

**Tools we used to do our Project:**

* [**Python3.**](https://www.python.org/)
* [**Pandas (Python Library).**](https://pandas.pydata.org/)
* **Pyswip (Python Library for Integration).**
* [**Google Colab (For Remote Code Execution).**](https://colab.google/)
* [**Code-OSS (Code Editor).**](https://flathub.org/en/apps/com.visualstudio.code-oss)
* [**Kali Linux (Operating System).**](https://www.kali.org/)
* [**Kaggle Website (For Our Data Set).**](https://www.kaggle.com/)
* [**Github (For Storing our Code).**](https://github.com/)
* [**Docker (For running dependencies Smoothly).**](https://www.docker.com/)

# **AI Detective Agent Report**

## **Table of Contents**

1. AI Detective Agent Overview

* Facts
* Rules
* Assert and Retract
* KB Structure

1. What Each Task Does

* 1. Find Crimes by Race (DFS)
* 2. Find Path Between Cities (A Star)
* 3. Choose Police Patroling Units (Genetic Algorithm)
* 4. Assign Police Units based on some limitations (CSP)
* 5. Check Crimes all the crimes happened in a City (BFS)
* 6. Find the best path for a City using heuristic only  (Greedy Best-First)
* 7. Find City with Most Crimes  (Hill Climbing)
* 8. Find Risky Cities by checking all possible options (MinMax)
* 9. Find Victim Race in a State using limit in DFS graph (IDDFS)
* 10. Find Risky Suspects by pruning the unnecessary (Alpha-Beta Pruning)

1. How We Make Sure that Assignment Requirements are complete.
2. All Done!

## **AI Detective Agent Overview**

Madam, I have created the agent for Lab Assignment No3. And integrate the agent with our Assignment no2 project. The agent can be run by running runme python file in the project directory. Which will present a main menu of all the possible functions of the agent. I have also integrated the prolog file of agent with our python file using the pyswip library. Now when we run the python code , it can communicate with our prolog KB and add and retrieve sentences or knowledge from it , as stated in the assignment requirement.

### **Facts:**

Madam, I have added the 50 facts in our prolog file according to the use case of our agent. I have also attached the suspects.csv file so that our agent can have a rich knowledge base.

### **Rules:**

Madam, I have added the rules to check and compare the facts and get answers. Eg a example, suspectForCrime checks certain crime parameters and returns the list of possible criminals.

### **Assert and Retract:**

### Madam, I have used assertz and retractall in every algorithm to manage facts in kb.pl. For each task, runme.py calls a Python script (like alphaBetaPruning.py) that uses assertz to add new facts, like assertz(suspectScoreFact(23, 27)), to save results. I used retractall to clear old facts, like retractall(suspectScoreFact(Age, \_)), to avoid duplicates. This happens in steps 4–5 for all tasks, ensuring the KB is fresh and accurate for queries.

### **KB Structure:**

Madam, our Kb has hardcoded 50 facts like neighboring cities (miami, chicago) and it is also integrated with suspects.csv file for having rich knowledge base facts . We also have the rules to compare and check the facts and find the useful information for our agent. Eg. our agent can check certain facts regarding a criminal using rule, and sort out the list of criminals based on certain criteria defined in the rule. It helps the agent to reason and make decision.

## **What Each Task Does:**

### **1. Find Crimes by Race (DFS):**

Madam, I have used Depth-First Search (DFS) to find crimes that are committed by special race people  , like “Black.” It searches through the complete data set and then find the crimes committed by black race people. OR you can type any other race in the parameter and serach it as well.

Madam, I have added certain :

* dfs.py to search suspects.csv for crimes commited by race.
* Stored some facts in prolog file like the murders commited by black race people has how many chances and severity.
* Certain rules to get the race related crime for the kb
* After the python program runs, it adds what it got result to the kb.
* Then it enquires kb about the special race people for the crime.
* Facts in kb.pl like raceCrimeFact(black, murder).
* A rule raceCrime(Race, Crime) to check crimes.

**The Screen Shot of Execution are attached below:**

### **2. Find Path Between Cities (A Star):**

Madam, I have used A Star to find the shortest path between cities, like Juneau to Bethel. It works on both heuristic(hn) plus actual cost f(n).

Madam, I have added:

* aStar.py to find paths using city data in our data set file US Crime Data Set.
* Facts in kb.pl like connectedCities(juneau, bethel).
* A rule in kb searchPath(Start, Goal, Path) to get paths.
* Then after program execution it got the result i.e the shortest path between the given cities.
* Then it add this result to the kb and our agent enquires the kb about the connected cities.

**The Screen Shot of Execution are attached below:**

### **3. Choose Police Patroling Cities (Genetic Algorithm):**

 Madam, I have used the Genetic Algorithm to pick the best patrolling cities based on the crime rate of different cities. We used the fitness function of crime in different cities to evaluate each patrolling suite.

Madam, I have added:

* geneticAlgorithim.py to select cities from US\_Crime\_DataSet.csv.
* Facts in kb.pl like deploymentCity(chicago).
* A rule optimalDeployment(City) to check cities.
* Then got the top fittest patrolling suite of the cities.
* Then add those cities to our Kb and then enquire the kb about the best crime patrolling suite.

**The Screen Shot of Execution are attached below:**

### **4. Assign Police Units (CSP)**

Madam, I have used Constraint Satisfaction Problem (CSP) to assign police units to high-crime cities, keeping it balanced.

Madam, I have used Constraint Satisfaction Problem (CSP) to assign police units to the different locations based on the crime type ratio and certain Constraints. Like Assigning the more units to the areas where there are more murders and less to the areas where are light crimes occur.

**Madam, I have added:**

* csp.py to assign units using crime data from our US Crime Data Set.
* Facts in kb.pl like policeAssignment(chicago, 5).
* A rule optimalAssignment(City, Units) to check assignments.
* Then we got the best fitted assignment of our police patroling based on certain constraints and resources like we always have limited number of cops.

**The Screen Shot of Execution are attached below:**

### **5. Check Crimes in a City (BFS):**

Madam, I have used Breadth-First Search (BFS) to look at crimes in a city, like Anchorage. It checks nearby cities step by step.

Madam, I have used the BFS to sort all types of crime that happens in a city be exploring each and every node possibility.

**Madam, I have added:**

* bfs.py to find crimes in US\_Crime\_DataSet.csv.
* Facts in kb.pl like cityCrime(anchorage, murder).
* A rule cityCrimeRelation(City, Crime) to list crimes.
* Then we got the list of all the crimes happened in a city .
* Then we add all those cities to the kb . And Enquire the KB about the all the crime happens in a particular city.

**The Screen Shot of Execution are attached below:**

### **6. Find City Path (Greedy Best-First):**

Madam, I have used Greedy Best-First Search to find a path between cities, like Anchorage to Juneau. It picks the city that seems closest.

Madam, I have used the Greedy Best First Search Algorithm to find the connected cities in using only the heuristic h(n).

**Madam, I have added:**

* greedyFirstSearch.py to find paths.
* Facts in kb.pl like connectedCities(anchorage, juneau).
* A rule searchPath(Start, Goal, Path) to get paths.
* Then we got the list of all the connected cities.
* Then I have stored it in neighboring cities and then enquire the kb about the connected cities.

**The Screen Shot of Execution are attached below:**

### **7. Find City with Most Crimes (Hill Climbing):**

Madam, I have used Hill Climbing to find the city with the most crimes. It picks better options to reach the worst city.

Madam, I have used the Hill Climbing to find the city with the most crimes. It got a city with crime number and just compare the upcoming cities about the number of crimes. It keeps comparing until it reaches a local minima or global minima.

**Madam, I have added:**

* hillClimbing.py to count crimes in US\_Crime\_DataSet.csv.
* Facts in kb.pl like mostCriminalCity(chicago, 100).
* A rule mostCriminal(City, Count) to find the worst city.
* Then it got the most criminal City as a result then it adds the city to kb
* Then Enquire the kb about the about the most criminal city.

**The Screen Shot of Execution are attached below:**

### **8. Find Risky Cities (MinMax):**

### 

Madam, I have used the MIn Mac cities to find the cities for police patrolling that is most suitable for the police or cops. Police are maximizers . They want to find the path that is most rewarding i.e crime eliminating for them. ON the other hand the criminals re minimizers they want to find the path that is most vulnerable for them. That is by using the path they can commit more crimes.

**Madam, I have added:**

* minMax.py to check risky cities in US\_Crime\_DataSet.csv.
* Facts in kb.pl like riskyCity(miami, 80).
* A rule risky(City, Score) to list risky cities.
* Then we got the maximum rewarding police patroling strategy as a result
* We add these results to the

**The Screen Shot of Execution are attached below:**

### **9. Find Victim Race in a State (IDDFS):**

Madam, I have used Iterative Deepening Depth-First Search (IDDFS) to find victim races in a state, like Black in Michigan. It searches deeper each try.

Madam, I have added:

* iddfs.py to check US\_Crime\_DataSet.csv for victim races.
* Facts in kb.pl like victimRaceFact(michigan, black).
* A rule victimRace(State, Race) to find races.
* Option 9 in runme.py to run IDDFS, save the race with assertz(victimRaceFact(michigan, black)), clear old ones with retractall(victimRaceFact(michigan, \_)), and decide, like “Talk to Black victims!”

**The Screen Shot of Execution are attached below:**

### **10. Find Risky Suspects (Alpha-Beta Pruning):**

Madam, I have used Alpha-Beta Pruning to find the most dangerous suspects by age, weapon, and victims. It skips useless data to pick the worst one.

Madam, I have added:

* alphaBetaPruning.py to score suspects in US\_Crime\_DataSet.csv.
* Facts in kb.pl like suspectScoreFact(23, 27).
* A rule suspectScore(Age, Score) to check scores.
* Option 10 in runme.py to run Alpha-Beta, save the score with assertz(suspectScoreFact(23, 27)), clear old ones with retractall(suspectScoreFact(\_, \_)), and decide, like “Watch suspects aged 23!”

**The Screen Shot of Execution are attached below:**

## **Why We make Sure that we meet the Assignment requirements:**

Madam, I have made sure the project meets the assignment:

* Used suspects.csv and US\_Crime\_DataSet.csv for data.
* Added 50 facts in kb.pl, like suspect details, scores, and city crimes.
* Wrote rules in kb.pl, like suspectScore/2, to check facts.
* Built runme.py with a menu for 10 tasks.
* Used assertz and retractall in every task to manage facts.
* Added checks to avoid crashes.
* Printed clear steps (1–6) for each task.
* Saved results to kb.pl and made decisions, like “Patrol this city!”

## **All Done:**

Madam, I have completed the AI Detective Agent. It works with two CSVs and Prolog for all 10 tasks. I used assertz and retractall to keep the KB updated. Screenshots will show it running.