RPM Project Report

Boyuan Liu
bryanliu@gatech.edu

1. INTRODUCTION

The Raven's Progressive Matrices are widely used in testing human intelligence. Therefore, they are also a good way to evaluate the intelligence of AI agents. An AI agent was built in this project to solve RPM problems using human-like heuristics. The agent's performance was evaluated using a set of 96 RPM problems. It solved 41/48 in the basic set, 35/48 in the test set, 15/48 in the challenge set, and 32/48 in Raven's set. The strengths and weaknesses of the AI agent were discussed in detail.

2. METHODS

The AI agent initially used OpenCV to detect contours and shapes in the images to solve RPM problems. However, the agent was unable to detect rotations of the shapes and also couldn't distinguish between shapes such as squares and rectangles. After examining the RPM problems in question sets C, D, and E, I decided to use pixel calculations in the more complex RPM problems. The images were pre-processed using Numpy and OpenCV. Images were first read using OpenCV as 2D arrays. Each image array was then converted to a 2D array of 0s and 1s with 0s representing white pixels and 1s representing dark pixels.

I created a series of functions that each representing a problem-solving heuristic. Each function examines if the RPM problem fits its criteria and if yes, returns a most likely answer and a confidence score. The agent will eventually output the answer with the highest confidence score as the final answer. The main heuristic functions are explained in the following section.

2.1 Check If Figures Are Identical

The agent will check if the figures in each row or columns are identical to each other. Comparing figure A and figure B is done by executing A - B and then using the np.countnonzero function to count non-zero pixels in the remaining array. If the result is 0, that means the two images are identical. However, sometimes the shapes in two identical images are aligned slightly differently. Therefore a threshold of 100 is set for this function. Any two images with less than 100 pixels difference will be considered identical. If the first row or column is made of all identical images, the agent will search through the answer set and find an answer that is identical to the other images in the same row or column.

2.2 Check If Rows or Columns Are Mirrored

In order to solve problems such as C-07 shown in Figure 1, the agent will use the np.flip function to mirror image A and compare it with image B. If image A and B are mirrored, the agent will then check if image B is the mirror of itself. The agent will also check if the same relation exists for image D, E, and F. If the agent determines that this function fits the problem, it will search in the answer set and find an answer that satisfies the same relationship.

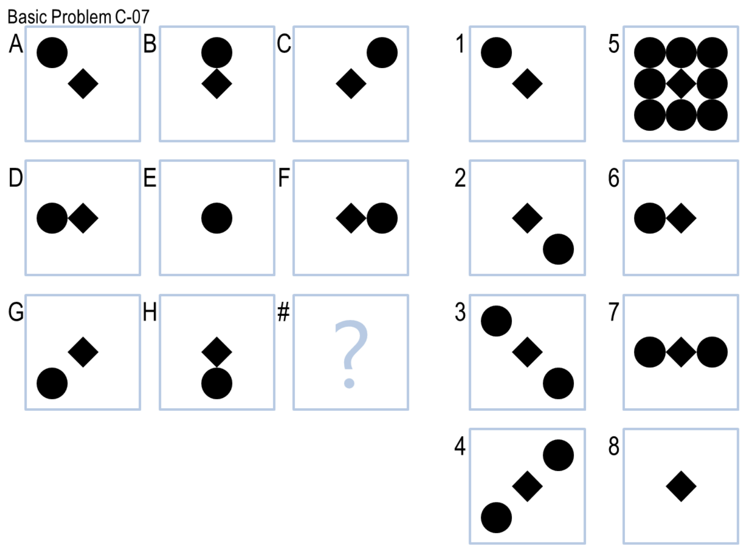

Figure 1-RPM problem example C-07.

2.3 Check If One Image Is The Combination Of The Other Two

As shown in Figure 2 below, some problems consist of images that are combinations of two other images. To detect this type of problem, image B and image C are combined by performing B + C. The function will then use np.clip to rescale the result to an array of 0s and 1s. This result will be compared to image A to check if they are identical. If the agent determines that this function fits the problem, it will use the same method to find an answer.

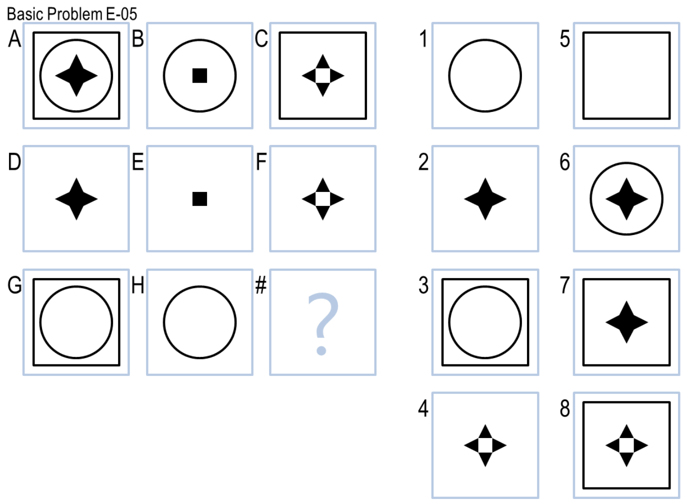

Figure 2-RPM problem example E-05.

2.3 Check If One Image Is The Common Area Of The Other Two

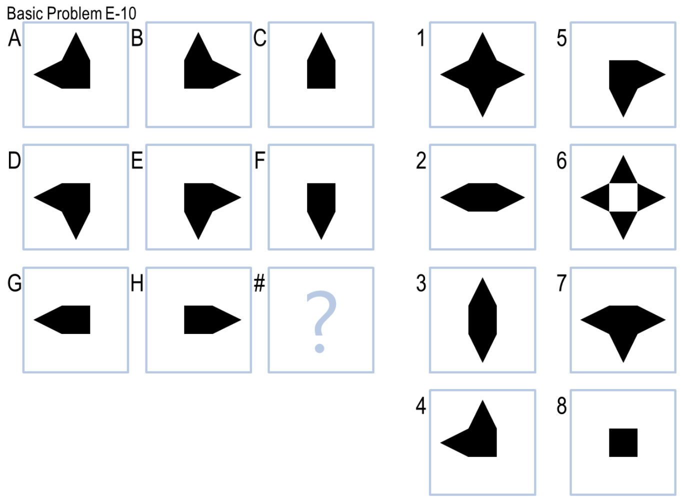

Figure 3-RPM problem example E-10.

For problems such as E-10 shown in Figure 3, the heuristic function will first perform A + B. The common areas will have a result of 2 and non-common areas will be 1. It will then remove all the non-common pixels by changing them to 0s and rescale the common areas from 2 to 1. The function then compares the result to image C. If they are identical, the function will find the answer that satisfies the same relationship in the answer set.

2.4 Check If Pixels Increase Or Decrease At The Same Rate

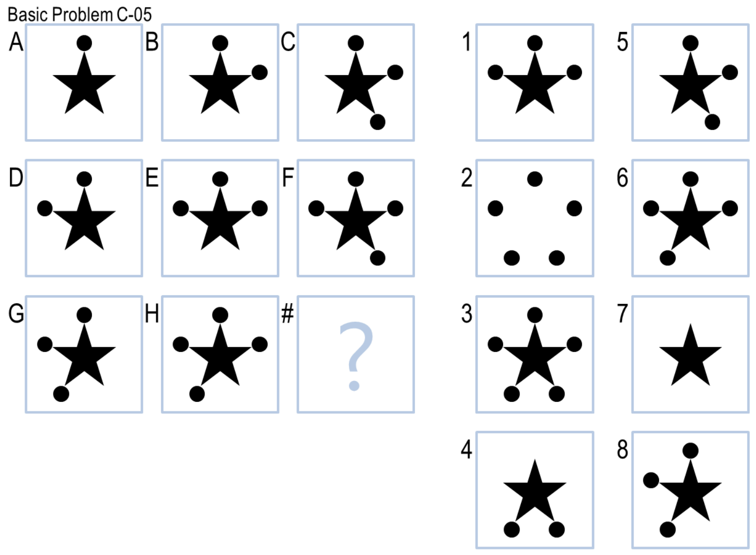

Figure 4-RPM problem example C-05.

For incremental RPM problems such as the one shown in Figure 1 above, I developed a function called method called method3x3_incremental. It calculates the incremental rate of pixels in the first two rows and checks if it can achieve a similar ratio in the third row with the answers provided. It uses the following equations to generation confidence scores for its answers:

$$\frac{ans-h}{h-g} \times 0.8 < \frac{c-b}{b-a} < \frac{ans-h}{h-g} \times 1.2 \ -> score\ 4$$

$$\frac{ans-h}{h-g} \times 0.9 < \frac{c-b}{b-a} < \frac{ans-h}{h-g} \times 1.1 \ -> score\ 6$$

$$\frac{ans-h}{h-g} \times 0.95 < \frac{c-b}{b-a} < \frac{ans-h}{h-g} \times 1.05 \ -> score\ 8$$

The letters in the equations, a, b, c, g, h, and ans represent the number of pixels in the corresponding figure. The idea is that the answer that achieves the closest incremental ratio receives the highest confidence score. This function works well with RPM problems that have a pattern of increasing or decreasing shapes.

The heuristic functions described above are the more general-purpose than can solve many different problems. There are also a few other functions that solve more specific problems such as comparing the total number of pixel in each row, and comparing the number of shapes in each image. The AI agent will evaluate the problem with each of these functions and output the answer with the highest confidence score.

3. SHORTCOMINGS

The agent was able to solve 41/48 in the basic set, 35/48 in the test set, 15/48 in the challenge set, and 32/48 in the Raven's set using the heuristic functions mentioned before. The agent solved the four test sets in 4.4 seconds. There are a few types of problems that the agent cannot solve. The agent was not able to solve problem D-08 shown in Figure 5 because it cannot recognize the white edges around certain shapes. The agent was only able to distinguish patterns of solid-filled and empty.

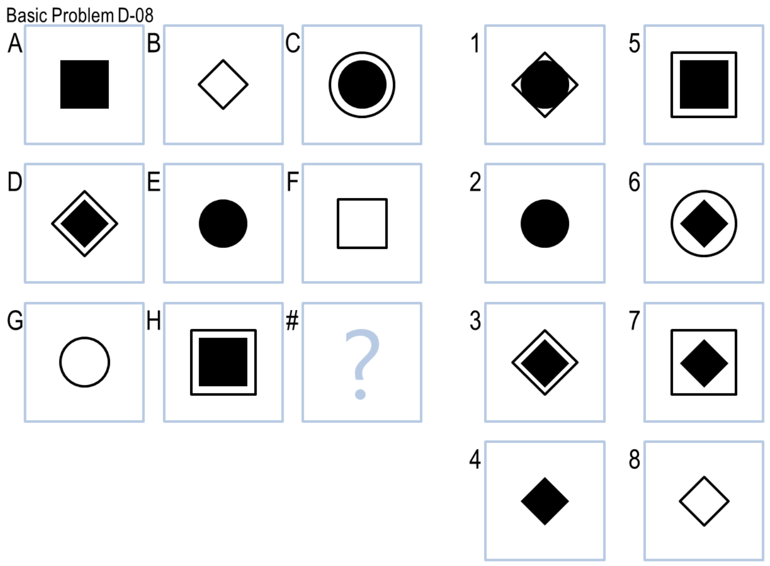

Figure 5-RPM problem example D-08.

The agent was also not able to solve problems such as C-09 shown in Figure 6. The agent can detect triangles and stars in the images. However, when processing images such as B, C, and H, the agent would label it as one complex shape instead of two triangles overlapping.

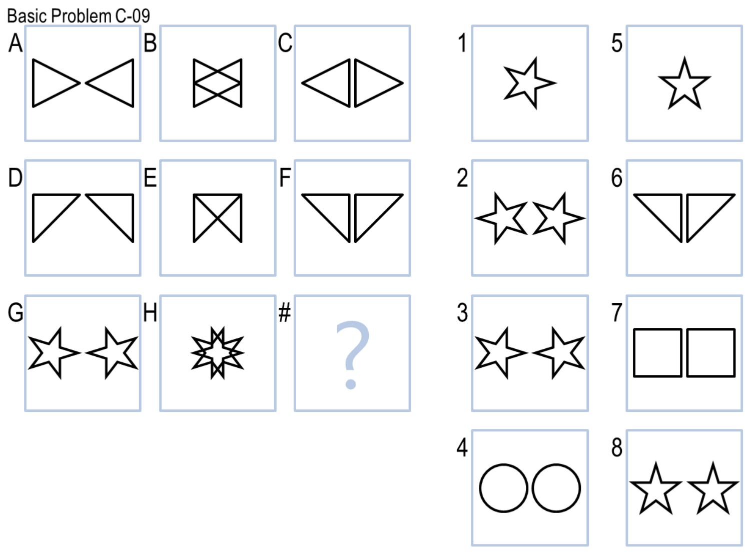

Figure 6-RPM problem example C-09.

4. AGENT DESIGN APPROACH

I initially designed the agent to iterate through a series of heuristic functions and output the answer with the highest confidence score. This structure has been working well as I add more heuristic functions to the agent. It is still used in the agent's final version. I have made a few minor changes to the scoring system throughout this project. In the final version, if the agent finds a heuristic that matches the problem perfectly and finds a perfect answer, it would stop the iteration and return the answer. If the agent determines that a problem is a close match to a heuristic, it would store the most likely answer and a confidence score based on how close the match was.

5. Agent Analysis

When people solve RPM problems, they usually have a few heuristics in mind. They would first determine if the problem can be solved by a heuristic they know of. If they do, they would solve the problem using the heuristic they know and if they don't, they would try to discover new heuristics to solve the problem. The AI agent solves the RPM problems by iterating through known heuristics, the same approach as humans, although it cannot discover new heuristics by itself like humans.

When it comes to individual heuristics, the AI agent solves some of them just like humans and some of them differently. For example, for problem E-05 shown in Figure 2 above, the agent would examine if image A and D are a combination of the two other images in the same row. It then searches in the answers set to find an image that satisfies the same relation. When the AI agent solves this type of problem, it behaves very similarly to humans. The agent solves problems different than humans in problems such as C-05 shown in Figure 4. The agent would calculate the increase in the number of pixels between each image and select '3' as the answer. However, humans can discover that each image has one more circle at the tip of the star than the previous image and select the same correct answer.