Li (Lily) Xu (lxu321@gatech.edu)

Solving Raven's Progressive Matrices problems with an KBAI agent
Project 1 Reflection

1. Introduction

Raven's progressive matrices was first published in 1938 and designed to measure one's mental ability to reason by analogy and form perceptual relations. RPMs are composed of 60 multiple choice questions, 5 sets with different difficulty levels, each contains 12 questions. The problems usually contain several panels of pictures with one missing. The test taker is supposed to pick up the best match of this missing piece from the provided candidates of answers. Since it is not verbal, RPMs can be used with persons from child to adult in any language. In this class, we will assess the algorithms and reasoning of our artificial intelligent agents by examining their performance against the RPM problems.

There are many potential difficulties about how to solve the RPM problems with an AI agent. For example, first of all, how does the agent start to understand a question? How can it recognize a circle, a triangle or any other element? Secondly, how does an agent build the concept of "transformation" or in another word, how does the agent decide on the changes and make correlations between elements before and after these changes? Moreover, how does an agent compare and rank different types of transformations and how does it find the best choice, if there is no perfect match? Here, I am trying to address some of the questions with what we learned from the first part of this KBAI class.

2. A brief summary of the agent

1. Data structure

The agent starts from the main method in RavensProject.java and reads in all the RPM problems. These problems are then stored in ProblemSet.java, which is composed of a list of Raven problems. Each Raven Problem is represented by RavensProblem.java, which consists of its problem name, problem type (such as "2x2" or "3x3") and a HashMap of RavenFigures. Each RavenFigure is composed of a figure name ("A"-"C" or "1"-"6") and the path of the visual file. The class RavenFigure also contains a HashMap describing attributes of Raven objects, but I won't be using this information as I intend to use purely visual method for this project.

With the path of a .pgn file, the program reads in the visual presentation of a RavenFigure:

```
RavensFigure A = problem.getFigures().get("A");
BufferedImage figureAImage = ImageIO.read(new File(A.getVisual()));
```

Then I implemented a method called `imageToArrayRGB` () to covert data stored in the BufferedImage into an int 2D array with 0 or 1:

```
int[][] A2D = imageToArrayRGB(figureAImage);
```

Every pixel in each row and column of the BufferedImage is iterated through, and converted to 0 (white) or 1 (black) based on its RGB value. For a grey pixel, a threshold was set so that it is considered as a black pixel if its value is above the threshold or otherwise a white pixel.

2. Methods implemented

to find the transformation between figures:

```
private String getTransformation(int[][] a2d, int[][] b2d, int[][] c2d)
```

to compare two transformations by scoring differences:

```
private int calculateDiff(int[][] a, int[][] b)
```

to find the choices closest to expectation

```
private int getAnswer(int[][] expect, int[][][] choices)
```

3. Result and analysis

Problem	Agent's Answer	Correct?	Correct Answer
Basic Problem B-01	2	Correct	2
Basic Problem B-02	5	Correct	5
Basic Problem B-03	1	Correct	1
Basic Problem B-04	3	Correct	3
Basic Problem B-05	4	Correct	4
Basic Problem B-06	5	Correct	5
Basic Problem B-07	6	Correct	6
Basic Problem B-08	6	Correct	6
Basic Problem B-09	**1**	**Incorrect**	**5**
Basic Problem B-10	3	Correct	3
Basic Problem B-11	1	Correct	1
Basic Problem B-12	1	Correct	1

Overall, I was able to solve most of the RPM problems in Basic Problem set B (11/12) within a few seconds. A few examples were chosen to illustrate how my agent solved these problems:

❖ **Identical figures**

For some simple RPMs, the figures in different panels are identical, as shown in Figure 1. By calling the method `calculateDiff` (A, B), my agent can compare the pixel-by-pixel the difference between two RavenFigures. Ideally, `calculateDiff` (A, B) should return 0. However, sometimes these figures may have slight shifts that are easily ignored by naked eyes. Thus I set a small number as a threshold (for example, 500 pixels), and anything pairs of images with a smaller difference should be considered as identical.

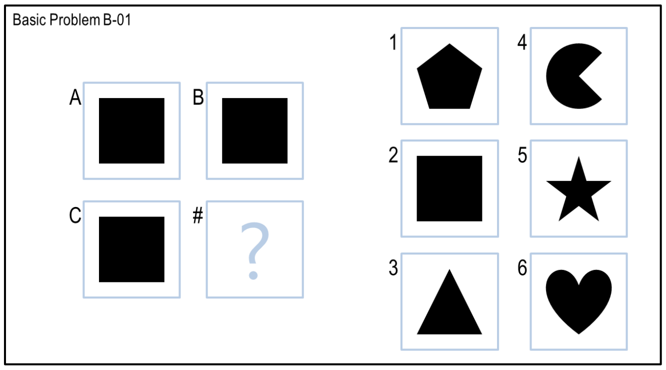


Figure 1

Thus, if a certain problem satisfies the criteria (A is identical to B or C), the transformation is labeled as "**identical AB**" (Figure 1) or "**identical AC**" (Figure 2). Then my agent will generate an answer based on existing panel and pick up the best matching answers from choices 1-6.

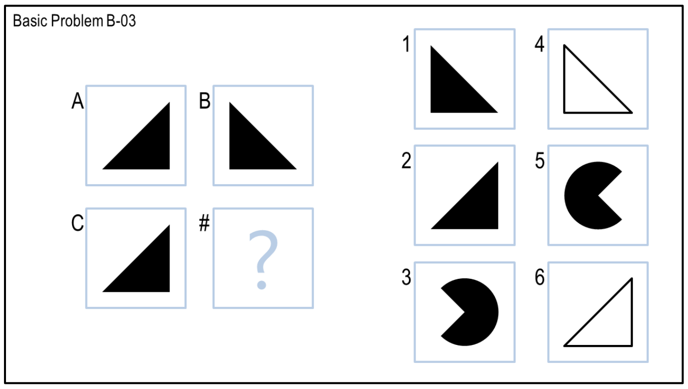


Figure 2

❖ **Symmetry**

Quite a few problems can be solved by symmetric transformations: either vertical symmetry or horizontal symmetry. My agent defined this transformation to be "**mirror_upDown_AC**" (Figure 3). So the agent will generate an answer based on panel B, and compare it to choices 1-6. Additionally, my agent can define similar transformation such as "**mirror_upDown_AB**" when figures are compared in the other direction, if horizontal symmetry exists between figure A and B.

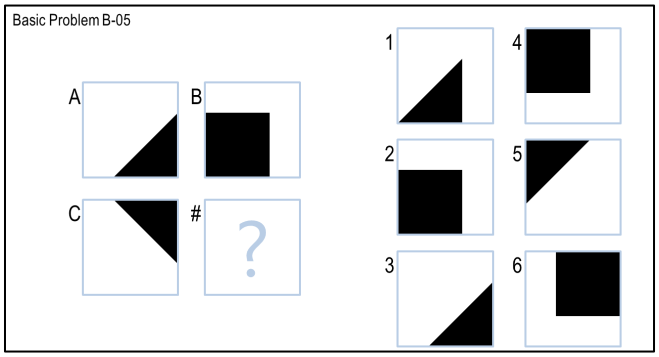


Figure 3 (horizontal symmetry example A→C)

Figure 4 is an example of the vertical symmetry transformation from A to C and named "**mirror_leftRight_AC**". Similarly the agent will generate an answer based on panel B, and compare it to choices 1-6. Additionally, my agent can define similar transformation such as "**mirror_leftRight_AB**" (Figure 5), if horizontal symmetry is detected between figure A and B.

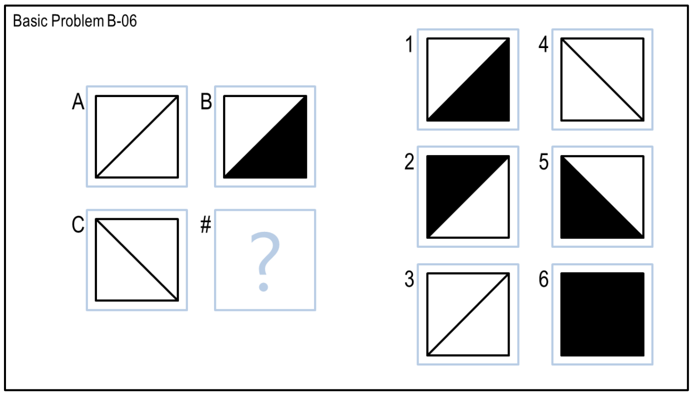


Figure 4 (vertical symmetry example A→C)

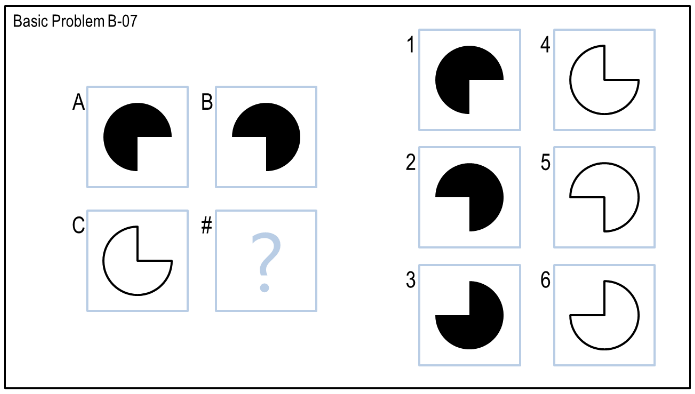


Figure 5 (vertical symmetry example A→B)

❖ **Counting Pixel**

Although it is not quite intuitive in human cognition, an AI agent can solve quite a few RPM problems by counting the number of filled pixels in each figure. An example is shown in figure 6, and the appearance of the small solid square leads to the increase of black pixel counts from panel A to C. Then the expected pixel numbers in "?" should can be calculated by pix(B)+ (pix(C)-pix(A)), and each choice answer will be examined to find the one with the closest number of pixels. Considering slight shifts of some figures, small differences should also be tolerated.

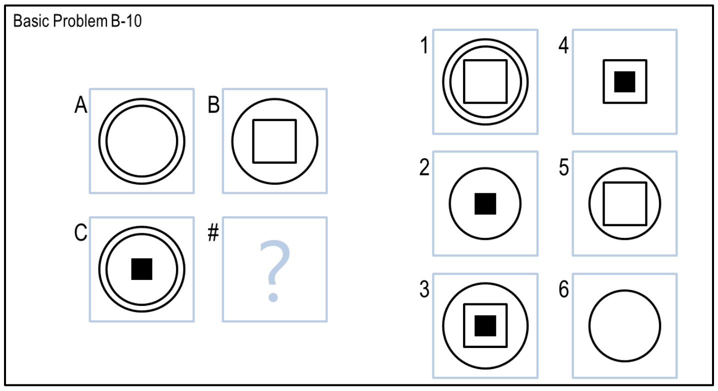


Figure 6

❖ **Problem my agent failed**

This AI agent seems to be quite successful as my first attempt to solve RPM problems with visual method. However, there are still many problems, which could potentially help to broaden and improve this original version in the following projects. For example, problems like shown in figure 7, cannot be solved correctly at the moment, as my rather simple agent is not able to make correlations between open and filled objects of the same shape. In the future, a combination of visual and verbal methods will most likely solve it.

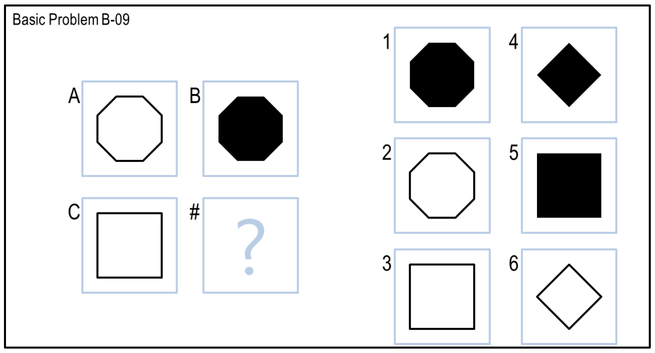
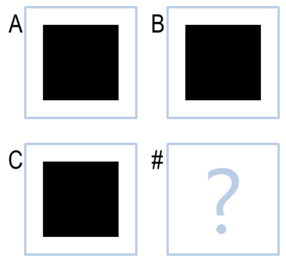


Figure 7(open and filled problem)

4. Discussion

Sometimes, a person can solve RPM problems with hidden rules that one does not even realize. For example, when we look at the answers, we pay attention to the most likely answer first. The computers can only iterate choices in order. Additionally, we constantly make logic deductions, generate hypotheses about correlations, then find violation to reject them/ evidence to confirm them. We gained these abilities most likely by receiving stimuli from the environment all the time. Our memory reinforces the experiences that we previously found stereotypes and helps us to better adapt to the nature. Thus the responses can be both helpful and illusive, but will not affect the reasoning of our computers.

For example, when we say two figures to be identical, for example panel A, B and C in figure 1, we use definitions such as shape, size, fill, color, position. These are features that we use to examine the world. However, when AI read in these pictures as 2D array, it strictly compared data at each pixel, and will not be able to compromise unless intentionally designed to do so, and these panels will be considered different even though they looks the same.



Another example, in Basic Problem B-04 (Figure 8), the transformation A--> C can be viewed as "horizontal symmetry" which is intuitive to human being and the answer is #3. However, this transformation can also be viewed as "rotating anticlockwise 90 degree", and the answer will become #1. Although a real intelligence will always pick #3, an AI will not have any preference without a definition of priority. In this way, we can teach an AI how to think like a human being.

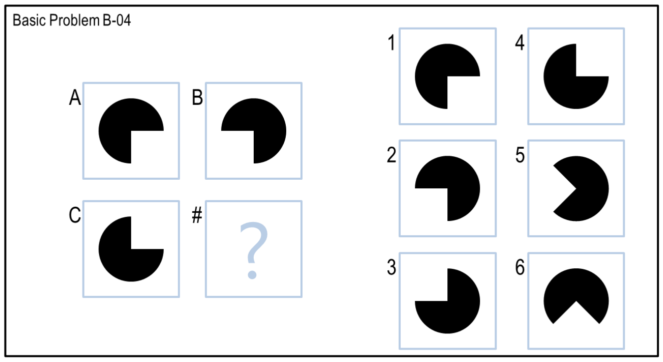


Figure 8