CS 7637: RPM Final Project Report

Teng Xue
txue34@gatech.edu

***Abstract*—This final report involves a thoroughly design, implementation, and evaluation for constructing an AI agent to conquer a whole RPM (Raven's Progressive Matrices) Project. A robust score mechanism and multiple general approaches that are targeting at tackling both 2x2 problems (Set B) and 3x3 problems (Set C, D, and E) will be demonstrated and discussed in detail.**

1 INTRODUCTION

The Raven's Progressive Matrices (**RPM**) test is a commonly used human intelligence test based on visual analogy problems (Raven, Raven, & Court 1998). RPM is regarded as a non-verbal estimate of fluid intelligence, it has not only been typically used to measure general human intelligence and abstract reasoning, but also has been often used as the psychometric measure of choice in educational and clinical settings.[1-2]

For copyright reasons, this project did not use actual RPM problems. Instead, this project used a set of **96 RPM-inspired problems** that was developed by Georgia Tech. While the individual shapes and their properties differ, these RPM-inspired problems mimic the same transformations and problem types as the actual standard and advanced RPM tests. [2] To be specific, these problems are broken into four problem sets: **2x2 Problem Set B (24), 3x3 Problem Set C (24), 3x3 Problem Set D (24), 3x3 Problem Set E (24)**. The individual problems capture the same reasoning as problems on the original RPM tests.

2 AGENT DESIGN

First of all, a key logic of my AI agent is to tell if two images are the same or not. Such logic is always the core step no matter what relations exist between two images. Since images are always stored as pixels, so my agent was designed to first convert all pixels in each image (Figure A-H and Option 1-8) into either white or black, followed by finding difference of two images using *ImageChops.difference()* by calculating and comparing white/black pixels ratio (**pr**) and black pixels density (**bpd**) for each image. Notably, even if two images look

exactly the same from us (humans), it still exists subtle difference due to the resolution of images and thus will never reach 100% match. Hence *threshold* was necessarily added to the agent to tolerate subtle difference between two similar images, the value of it should be kept as small as possible to exclude dissimilar images. Thus, if the **pr** of the difference of two images is smaller than the *threshold*, the two images will be considered same.

My AI agent includes two solvers: **"2x2 problem solver"** (focused on Set B) and **"3x3 problem solver"** (focused on Set C, D, and E). An integrated **score mechanism** was employed inside of each solver to help the agent making the final choice. Specifically, after applying this score mechanism to all options by the addition of all individual scores, the agent would return an option with the highest score as the final answer.

3 AGENT IMPLEMENTATION

3.1 Tackling 2x2 Problem Set B

For 2x2 RPM problems, only 2D relations exist between images at most (horizontal AB and/or vertical AC). Hence only four individual scores were included in the **score mechanism** of "2x2 problem solver": *"reflection_score"*, "*rotation_score"*, *"difference_score"*, *"imageFill_score"*.

- **Reflection_score**: the agent would compare Figure A with both Figure B and Figure C. If a reflection relation (unchanged/ left-right/ top-bottom) is found, the agent will give the option which matched the same relation with Figure C (if AB) or Figure B (if AC) the highest score (**2 points**). This logic was implemented by using *Image.transpose(image.type)* where "type" could be either *FLIP_LEFT_RIGHT* or *FLIP_TOP_BOTTOM*.
- **Rotation_score:** the agent would compare Figure A with both Figure B and Figure C. If a rotation relation (exclude reflection) is found, the agent will give the option which matched the same relation with Figure C (if AB) or Figure B (if AC) the highest score (**1 point**). This logic was implemented by using *Image.rotate(degree)* where "degree" was set to 270 (clockwise 90 degrees).
- **Difference_score:** the agent would compare Figure A with both Figure B and Figure C. If a difference relation (addition/subtraction) is found within a *threshold* of 3.7%, the agent will give the option which matched the same

relation with Figure C (if AB) or Figure B (if AC) the highest score (**1 point**). This logic was implemented by using *ImageChops.difference(image 1, image 2)*.

- **ImageFill_score:** the agent would compare Figure A with both Figure B and Figure C. If an image filling relation (turn a white shape into a black shape) is found, the agent will give the option which matched the same relation with Figure C (if AB) or Figure B (if AC) the highest score (**5 points**). This logic was implemented by using *ImageDraw.floodfill(image, xy = center, value = 0)*.

3.2 Tackling 3x3 Problem Set C, D, and E

For 3x3 RPM problems, more complex 3D relations could exist between images including horizontal (ABC), vertical (ADG), and diagonal (AE#) relations. Therefore, six individual scores were well-designed and calculated in the **score mechanism** of "3x3 problem solver": *"match_score"*, *"union_score"*, *"reflection_score"*, *"rotation_score"*, *"symmetry_score"*, *"difference_score"*.

- **Match score**: the agent would compare Figure A with all option figures (1-8). If an identical relation (unchanged) is found, the agent will give the option which matched Figure A most the highest score (**1 point**).
- **Union score**: the agent would calculate and union black pixel density (**bpd**) of each image in the first two rows. If a total **bpd** of the 1st row is close enough to the 2nd row within a *threshold* of 0.2%, the agent will give the highest score (**10 points**) to the option which makes a total **bpd** of the 3rd row also close to the first two rows within a *threshold* of 0.2%. If a *threshold* higher than 0.2% but within 1.2% has to be used, the agent will give that option **8 points** instead.

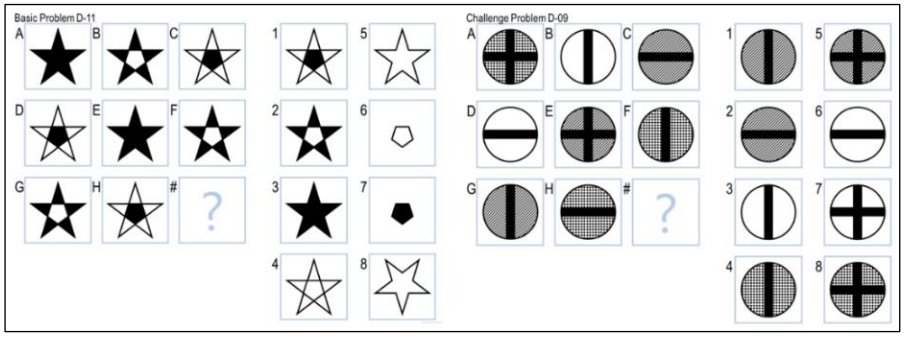

Figure 1— Selected Basic Problem D-11 and Challenge Problem D-09 were solved by using "union score" logic.

For example, in **Figure 1** above, a common property of these two problems is that all three images between each row are the same, either the whole image

(D-11) or parts of the image (D-09), but just changed position. In other words, a total number of black pixels between each row should be identical no matter what the shape is. For black pixels: row1 = (A+B+C); row2 = (D+E+F); row3 = (G+H+#). So if row1 = row2, then the agent will expect row3 = row1 and row3 = row2, thus finding the image of # from row3.

- **Reflection score**: the agent would compare Figure A with Figure B. If a reflection relation (left-right) is found and white/black pixel ratio (**pr**) of Figure C is greater than **pr** of Figure A within *threshold* of 12%, meaning C is an overlap image of A and B, then the agent will give the option whose **pr** is also greater than **pr** of Figure G within *threshold* of 12% the highest score (**15 points**) (*e.g.*, E-11 in **Figure 2**).
- **Rotation score:** the agent would compare Figure A with both Figure C and Figure G. If a rotation relation (exclude reflection) is found, the agent will give the option which matched the same relation with Figure G (if AC) or Figure C (if AG) the highest score (**8 points**).
- **Symmetry score**: the agent would firstly split Figure A, B, and C into two halves by vertically (*left_A/B/C; right_A/B/C*) and horizontally (*top_A/B/C, down_A/B/C*). Then checking if *left_A = right_C* and *right_A = left_C*, **or** *top_A = top_C* and *down_B = down_C*. If the former relation is found, meaning A and C are symmetry in left/right, then the agent will split Figure G and all 8 options vertically into two halves too, then giving the option which matched the same relation with Figure G the highest score (**5 points**). **Or** if the latter relation is found, indicating A, B and C are symmetry in top/down, then the agent will split Figure G, H, and all 8 options horizontally into two halves too, then giving the option which matched the same relation with Figure G and Figure H the highest score (**5 points**) (*e.g.*, E-09 in **Figure 2**).

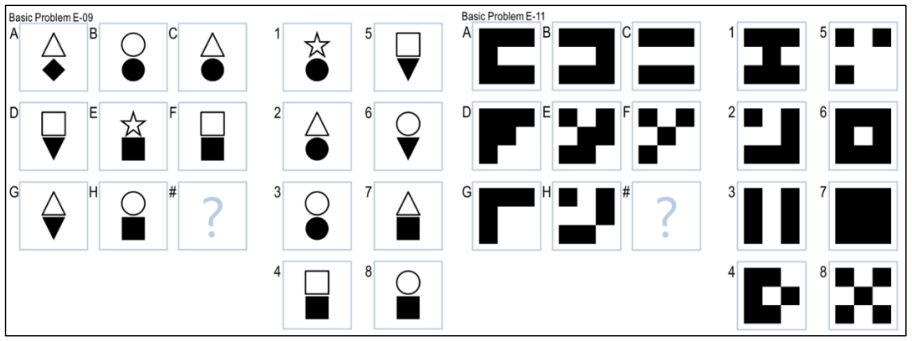

Figure 2 — Selected Basic Problems E-09 and E-11 were solved by using "symmetry score" and "reflection score" logics.

- **Difference score_general:** the agent will first compare white/black pixel ratio (**pr**) between Figure C and F to search a **vertical/column relation**, then compare **pr** between Figure G and H to search a **horizontal/row relation**. For example, in **Figure 3 below**, started from the vertical/column relation search:
    1. If **pr** of C is higher than **pr** of F within a *threshold* of 23%, meaning **pr** decrease (black pixel increase) with column. Then the agent will compare a difference between C and F (**diff1**), as well as a difference between F and an option figure (**diff2**).
    2. Next checking if **pr** of F is higher than **pr** of that option figure within a *threshold* of 40%, meaning that option figure matched the relation as **pr** keep decreasing in that option figure (more black pixels as expected). Then the agent will compare the similarity of **diff1** and **diff2**, assign different score to different degree of similarity: **15 points** for 96% similarity, **10 points** for 94% similarity, and **5 points** for 92% similarity, etc.
    3. Instead, if **pr** of C is lower than **pr** of F within a *threshold* of 27%, meaning **pr** increase (black pixel decrease) with column. Then the agent will compare a difference between C and F (**diff1**), as well as a difference between F and an option figure (**diff2**) which is same as step 1.
    4. Next checking if **pr** of F is lower than **pr** of that option figure within a *threshold* of 40%, meaning that option figure matched the relation as **pr** keep increasing in that option figure (fewer black pixels as expected). Then the agent will compare the similarity of **diff1** and **diff2**, assign different score to different degree of similarity which is same as step 2.
    5. After finished the vertical/column relation search, then proceeding the horizontal/row relation search by repeating step 1-4 for Figure G and H.

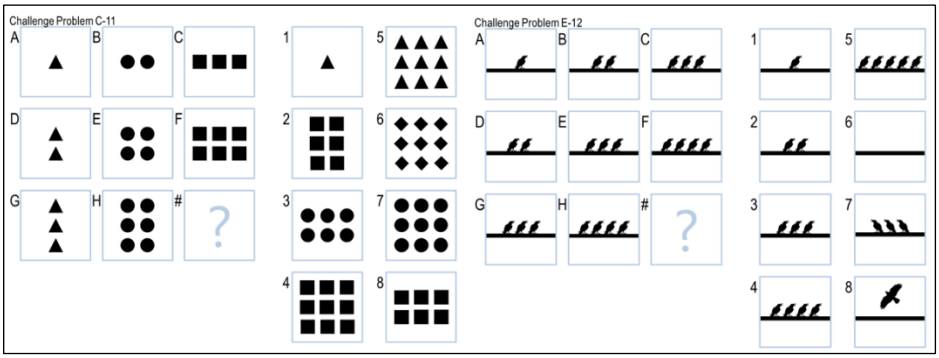

Figure 3 — Selected Challenge Problems C-11 and E-12 were solved by using "Difference score_general" logic.

- **Difference score_setE**: the agent would firstly compare the difference between Figure A and Figure B to generate a **diffo** image (*diffo = ImageChops.difference(A, B)*). If image A = B, diff1 should be completely black or so, because the rgb value of black is {0,0,0}. So the agent will has to convert white/black pixel of **diffo** to form a **diff1** image (*diff1 = ImageChops.invert(diffo)*). Similarly, a converted difference image between Figure G and H is generated as a **diff2** image using the same method. Then the agent will compare if **diff1** is close enough to Figure C by comparing their white/black pixel ratio (**pr**) within a *threshold* of 4%. If such relation is found, indicating A+B=C (*e.g.,* E-02 in **Figure 4**) or |A-B|=C (*e.g.,* E-07 in **Figure 4**), then the agent will give the option which also close enough to **diff2** the highest score (**15 points**) by comparing their **pr** within a *threshold* of 6%.

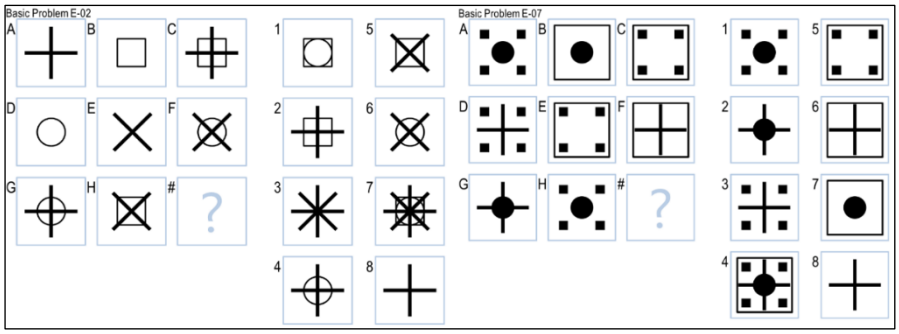

Figure 4 — Selected Basic Problems E-02 and E-07 were solved by using "Difference score_setE" logic.

4 AGENT PERFORMANCE

Basic Problems B: 12/12; Basic Problems C: 8/12; Basic Problems D: 9/12; Basic Problems E: 10/12.

Challenge Problems B: 7/12; Challenge Problems C: 5/12; Challenge Problems D: 5/12; Challenge Problems E: 7/12.

Overall, my AI agent performed well on most of Basic Problems and also passed a half of Test Problems, Challenge Problems, and Raven's Problems, respectively, with a fair runtime that shown from the *Gradescope* (**Figure 5**). It only cost my agent around **30 seconds** for solving all **192 problems** (48x4), and ultimately returned **112 out of 192** problems a correct answer. The algorithmic complexity of my agent is O(n) where n is the number of option figures.

```
Software:

        Python version: 3.8.0
        NumPy version: 1.19.5
        Pillow version: 8.1.0
        OpenCV version: 4.2.0


Results:

        Basic passed: 39, failed: 9
        Test passed: 25, failed: 23
        Challenge passed: 24, failed: 24
        Raven's passed: 24, failed: 24
        Runtime: 30.414998769760132 seconds
```

Set	Correct	Incorrect	Skipped
Basic Problems B	12	0	0
Basic Problems C	8	4	0
Basic Problems D	9	3	0
Basic Problems E	10	2	0
Challenge Problems B	7	5	0
Challenge Problems C	5	7	0
Challenge Problems D	5	7	0
Challenge Problems E	7	5	0

Figure 5— Submission result from the Gradescope and running result from the local test.

5 AGENT EVALUATION

5.1 Limitations of the Agent

My AI agent did **struggle** on some certain type of problems such as Basic Problems D-08, D-12 and Basic Problems E-04, E-12. This was because most of them involved **more than one relation** between images which made some conflict and confuse to my agent's determination. And currently there is still **no ideal logic** exist in the agent to deal with this issue. For instance, D-08 not only involves shape change relation but also has size change and color change relation. D-12 even includes an extra relation of the number change. E-04 includes both size change relation and position change relation. E-12 has relation of the number change as well as an extra relation of orientation change (**Figure 6**).

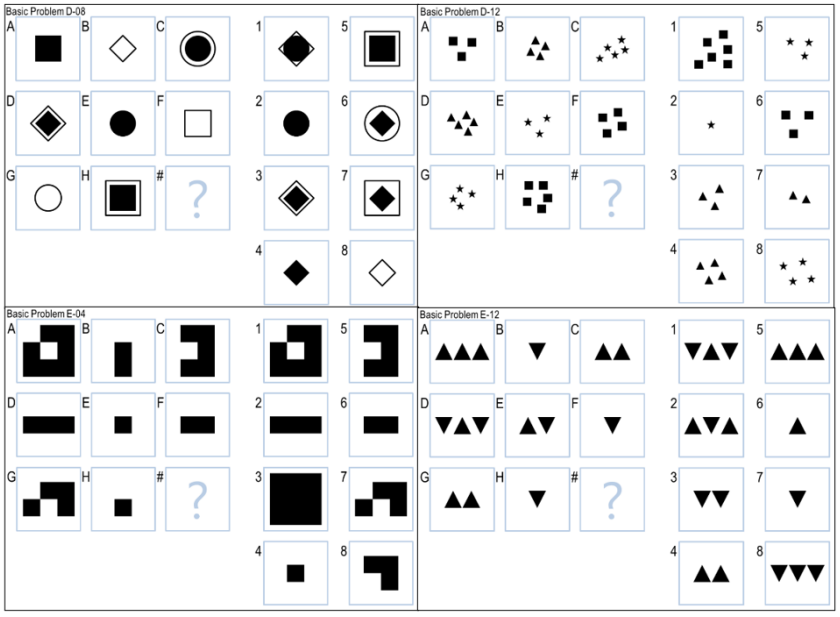

Figure 6— Selected failed Basic Problems D-08, D-12 and E-04, E-12 involve multiple relationships between images.

5.2 Concept Employment in a Knowledge-based AI Agent

My AI agent did employ multiple concepts (knowledge) that learned from CS 7637 class such as **Case Based Reasoning** (*e.g.*, retrieval of individual scores and storage in the score mechanism), **Incremental Concept Learning** (*e.g.*, generalize and specialize the corresponding threshold value), **Constraint Propagation** (*e.g.*, different affine transformations), and **Classification** (*e.g.*, "2x2 problem solver" and "3x3 problem solver"), etc.

Most importantly, the agent was governed by the **Generate & Test** method. A smarter generator was designed to only generate relevant figure to compare, and a smarter tester was designed to only test relevant relation. Taking a 2x2 RPM problem as an example, if a relation of 90-degree rotation was found between Figure A and Figure B (**AB**), a smarter generator would only generate Figure C to compare with option figures (**C-op**); a smarter tester would only apply the same relation of 90-degrees rotation to **C-op**.

5.3 Agent *vs.* Human

My AI agent is **not similar to human's approach** to tackle these RPM problems. As I firstly described, my agent will make a choice based on the outcome from the score mechanism by calculating and comparing white/black pixels ratio (pr) and black pixels density (bpd) for each image. Apparently, however, human won't compare pixel ratio and score all options to make a choice. The principal reason behind this is that human could differentiate shapes inside of each image, while the agent could only recognize pixels.

Nevertheless, the inherent logic of all individual scores that calculated by my agent is exactly **same as human's reasoning**. Since the agent was designed by me (human) to mimic my reasoning process. For instance, both the agent and human would find reflection/rotation relations by looking for affine transformations; both the agent and human would find symmetry relations by splitting images; both the agent and human would find addition/subtraction relations by comparing the difference between two images, etc.

6 REFERENCES

1. Bilker, Warren B.; Hansen, John A.; Brensinger, Colleen M.; Richard, Jan; Gur, Raquel E.; Gur, Ruben C. (2012). Development of abbreviated nine-item forms of the Raven's standard progressive matrices test. *Assessment*. *19* (3), 354–369.
2. Joyner, D., Bedwell, D., Graham, C., Lemmon, W., Martinez, O., & Goel, A. (2015). Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests. In *Proceedings of the Sixth International Conference on Computational Creativity*. Provo, Utah.