RPM Final Project Report

Muhammad Uzair Shahid Syed
msyed46@gatech.edu

1 HIGH LEVEL AGENT FUNCTION

1.1 Agent approach

The agent solves the Raven's test of intelligence using knowledge representation and generate and test methods. It generates the knowledge representation stored as frames of the images (A, B, C etc.) and then finds the closest possible answer (images 1, 2, 3 etc) that aligns with the learnt knowledge by testing for consistencies in relationships and patterns it has identified in the frame. In some cases, the agent generates a possible solution from the knowledge frames, then tests the possible answers against the generated solution and seeks to find an answer which abide by the relationships identified. In cases where the agent thinks there are more than one possible answer, the agent uses the method of problem reduction to find the most closest answer.

The methods used by the agent to populate the knowledge representation are pixel comparisons, dark pixel ratios (DPR), transformations, bit-wise operations and shape detection.

1.2 Functionality: Overview

The agent initially identifies whether the set of problem is a 2 by 2 or a 3 by 3 matrix. Based on that, the agent uses different functions but similar approach. This is because the agent doesn't need to loop through images A to H in the case of 2 by 2 matrix and it can be more efficient in finding an answer.

In either 2 by 2 or 3 by 3 matrix, the agent utilizes the following methods to populate the frames:

1. Identifying transformation - exact, rotated or mirrored images
2. Identifying pixel changes and average pixel change across row or diagonally
3. Identifying the shapes, number of shapes and orientation of shapes within the image
4. Identifying dark pixel ratio between two images in a row or a diagonal

Next, the agent uses the populated knowledge frame to identify an answer. For

a 2x2 problem set, the agent checks the logic in the following ordered manner:

- Function 1: Transformation check
- Function 2: Row relationships check - pixel change/DPR

For a 3x3 problem set, in addition to the above two logic, the agent checks for additional logic in the following ordered manner:

- Function 3: Diagonal relationships check - pixel change/DPR
- Function 4: Bit-wise operations check - performed row-wise or column-wise
- Function 5: Shape detection check

All of the above logic are stored as separate functions and in the given order. If an agent cannot find an answer through transformation check, it moves to the second function and so on until it finds a result. If it finds a result, it will skip the other functions to avoid further complication. Bit-wise operations are not stored in the frames - instead if the first three functions fail to result in an answer the agent attempts to perform a bit-wise operation using generate and test method. The shape detection method is the last one to be considered due to complexity and computation time required in detecting individual shapes and any existing patterns.

1.3 Function 1: Transformation check

The transformations the agent checks for are rotation, mirrored images or exact images. The agent always first attempts a row-wise check before moving onto column or diagonally. If the images in a row are exact, and in the case of a 3x3 matrix the images in the second row are also exact then the agent is looking for an answer which will keep this consistent for the third row.

Similar logic applies for mirrored or rotated images. For example in *Appendix: Figure 2*, A and B are mirrored images therefore the agent attempts to find a mirror image of C to keep consistent with the knowledge frame.

1.4 Function 2: Row relationships check

In the row relationships function, the agent is using either pixel changes between images (or average pixel changes for 3x3 matrices) or DPR from the knowledge frame to find a potential answer. This section only covers a few ways the agent uses these attributes in a row to find an answer. The agent was built in with more logic than what is described here but the essence is similar.

For example, the agent would attempt to see if the average pixel change in both row 1 and row 2 of a 3x3 matrix are very similar with a threshold of 5 percent similarity. If that is the case, the agent attempts to find an answer and tests it by ensuring the average pixel change between row 1, row 2 and row 3 are all within 5 percent.

Furthermore, if the agent identifies the absolute value of the pixel changes between AB and BC in a 3x3 matrix is within an 8 percent similarity, but their raw values are opposite signs then it is possible the images are moving inward or outward in a consistent manner. No additional pixels are being added rather the movement and intersection of pixels is causing a consistent change in pixels. If the same relationship is seen in the second row, then the agent finds an answer and tests it to ensure the same relationship is detected in the third row.

An example for a 2x2 matrix can be done by looking at *Appendix: Figure 3*. In this case the agent has learnt by populating the frame about what each shape in the images are, the number of sides and the pixel changes in a row or column. Using this information, the agent can understand that from A to B the number of sides are same but the pixels have increased. Therefore, the agent finds an answer which matches this understanding most closest by picking a black-filled square.

Similar threshold percentage for similarity are used for dark pixel ratios.

1.5 Function 3: Diagonal relationships check

Similar to row relationships, the diagonal relationship uses same principles and thresholds to find the answer. However, this time the agent checks DPR and pixel changes in a diagonal fashion.

For example, if pixel change or DPR relationships between A to E and B to F is similar, and D to H and F to G is similar then we can deduce that the E to answer will be similar to D to H. Also, if there is no pixel change between A to E, B to F and D to H, then we can deduce that all diagonal images are the same.

Depending on what it learns, the agent would find an answer and test it to ensure the diagonal relationship identified is present.

Other diagonal relationships are present as part of the next two subsections.

1.6 Function 4: Bit-wise operations check

In the bit-wise operation function, the agent finds an answer using logical operators AND, OR, XOR, or pixels combinations within the rows. Some of the combinations that are used by the agent are:

- Pixels in (A - B) = pixels in C, and pixels in (D - E) = pixels in F. Therefore, agent finds an answer by using pixels in (G - H) to equal pixels in answer
- Pixels in (A + B) = pixels in C, and pixels in (D + E) = pixels in F. Therefore, agent finds an answer by using pixels in (G + H) to equal pixels in answer
- Pixels in (B + C) = pixels in A, and pixels in (E + F) = D pixels. Therefore, agent finds an answer by using pixels in (H + answer) to equal pixels in G
- Image A AND/OR/XOR image B equals image C, image D AND/OR/XOR image E equals image F. Therefore, agent finds an answer by using image G AND/OR/XOR image H to equal answer

Other combinations between the images and logical operators have also being applied to capture precision in the agent's answer.

1.7 Function 5: Shape detection check

Lastly, if none of the above attempts find the agent an answer, the agent starts to detect shapes to understand relationship of shapes in each row. The agent creates a separate frame for each image which tells the agent how many squares, triangle, circle, stars or number of shapes exist in the image.

Then, the agent checks for repeating patterns of shapes and if there is one, then what shape should the answer be to continue this pattern. There is a separate function to determine shape pattern recognition. For example, if a given problem has the first row as [triangle, square, circle], the second row as [square, circle, triangle] and the third row as [circle, triangle]. The agent is able to identify that the third image in the third row should be a square. The agent will then attempt to find if the number of shapes in each image in a row also has a pattern and based on this information, the agent is able to find an answer.

1.8 Finding the answer and voting

Based on the explanations given above, the agent first understands relationships between images in a row or diagonal approach using DPR/pixel density/bit-wise operation/shape detection. It stores this information in a knowledge frame

and uses that information to find a possible candidate for an answer. In some cases, the agent generates a possible solution, looks for a candidate that is the same as the generated solution and test it against the knowledge frame to ensure relationships are consistent.

In each of the above methods of finding an answer, if the agent finds more than one answer it stores it in an array. The agent will remove any duplicate answer it has added to the array. The agent will attempt to vote for the best possible answer that provides a better match and abides by the relationships.

For example, the agent could try to select an answer which has an exact match to images A and E in diagonal relationships, or the agent could try to make sure the possible answer is not similar to any of the images from A to H. The voting system in my agent is not perfect, but is able to help with some cases where the agent is unable to best identify one answer.

2 OVERALL PERFORMANCE

Agent performance can be defined as how reliable the agent is in using the same methods described in section 1 to find the correct answers. For example, if the agent can perform well on 3 known problems it was programmed against, can it then use the same methods to come to the correct answer with unknown test cases?

	Basic	Test	Challenge	Raven's	Set Combined
Set B	12	10	5	9	36
Set C	12	8	5	7	32
Set D	10	5	2	2	19
Set E	9	7	3	6	25
Total	43	30	15	24	

Figure 1—Final result of the RPM agent

2.1 Definition and statistics

From Figure 1, it can be seen that the agent passes 43 Basic, 30 Test, 15 Challenge and 24 Raven's test on Gradescope. The agent is able to achieve a final score of 73/96.

As per the definition of performance above, I believe the agent is quite good in

it's performance. If we look at the specifics, I think the agent performs extremely well in Basic set overall compared to Test set. It performs poorly on all Challenge sets with a score of 15/48 which means the agent would require further tweaking. Another way to look at the results is that the agent's performance is less on Set D (Basic/Test/Challenge/Raven's combined) with only 19/48 score as compared to the other sets. This is due to the troubles I was facing when trying to generalize the agent when attempting Set D problems.

Overall, I am very happy with the performance of my agent in Basic, Test and Raven's set. The fact that the agent was built and programmed against the known Basic set only and results in more than 50 percent score (30/48) in Test and exactly 50 percent (24/48) on Raven's indicates a really good performance.

3 EXAMPLES OF GOOD PERFORMANCE

In this section, I will only be using the Basic sets as examples where the agent performed really well. For the images referenced in each section, please refer to the Appendix.

3.1 Problem C-11: Center of mass of dark pixels

An example of good performance is problem set C-11 which can be seen in *Appendix: Figure 4*. The agent learns that the answer needs to have three diamonds by looking at how each image in the row from left to right increases by one diamond. Knowing image H has 2 diamonds the agent needs to look for an answer with 3 diamonds. The problem is that both image 3 and 4 have three diamonds each. The agent chooses them both as possible answers and moves onto voting for the best answer.

When voting, the agent attempted to look at the center of mass of the dark pixels in image G and H in an x and y co-ordinate. If image G and H center of mass y-coordinates are the same but x-coordinate changes, that means the answer should have similar y-coordinates (5 percent threshold) regardless of the x-coordinate change. The reason being the image G and H have diamonds with same center of mass y-coordinates but the x-coordinate would change due to addition of another diamond. Therefore, adding a third diamond should change the x-coordinate of the center of mass but should be aligned at the same y-coordinate. Therefore the agent is able to pick the answer image 4.

3.2 Problem D-02 and D05: Diagonal relationships

In problems D-02 and D-06, the agent is able to identify a diagonal relationship is present using the knowledge it has learnt when analyzing the images A to H. Furthermore, the agent is able to distinguish whether to approach the diagonal relationship with a DPR method (D-02) or simply calculating pixel changes (D-05) with a 5 percent threshold. This was very crucial to performance because now the agent can understand what approach to apply for similar situations. This was a primary reason for a decent score in Basic set D. Please refer to *Appendix: Figure 5* for problem D-02.

3.3 Problem Set E: Bit-wise operations

In Set E, the agent performed well by using mostly bit-wise operators such as AND, OR and XOR logic for all basic set E problems except E-09 and E-12 under 0.1 seconds each. From *Appendix: Figure 6*, it can be seen that the agent can simply perform A OR B to find C and D OR E to find F. The agent did well because once it learns that a bit-wise operation can be used, the agent applies the same bit-wise operator on image G and H to generate a possible answer and test it against answers 1 through to 8 until it finds an exact match.

4 EXAMPLES OF AI STRUGGLES

In this section, I will only be using the Basic sets as examples where the agent performed poorly. For the images referenced in each section, please refer to the Appendix.

4.1 Problem D-08: Shape detection

The agent struggled in cases such as D-08 in *Appendix: Figure 7* where a diagonal relationship exists and the shape changes from A to E. The agent enters the shape detection function explained above and is able to identify image A having a filled shape with 4 sides and image E being filled with many sides essentially being a circle. When the agent loops through the possible answers, its able to see there are 6 images with 4 sides and is unable to identify a diamond from a square, as well as a shape within the diamond or square. The agent essentially struggles in identifying a strong pattern and a possible answer when the answer options are very similar and have complicated shapes.

4.2 Problem D-10: Pattern identification

The agent also struggled with problems such as D-10 in *Appendix: Figure 8* where it is unable to identify any pattern from the current agent operation. Even as a human, I have a hard time figuring out what the answer to this problem set would be. The agent cannot determine the answer using pixel change/DPR method, unable to find concrete relationship in a row or diagonal viewpoint, not successful with bit-wise operations and even though it detects the shapes in each images the agent is unable to identify a pattern. Thus, D-10 was a strong pain point for the agent and myself!

4.3 Problem E-09: Multiple shapes and patterns in a row

Another example of where the agent struggled was in E-09 in *Appendix: Figure 9*. This was also something I found hard to find an answer for. However, after some time I was able to see that there is a pattern in a row where the top shape for image A is same as image C and bottom shape in image B is same as image C. This pattern is consistent with the second row and using the logic gives the correct answer. However, the agent is unable to find the answer because this particular logic with multiple shapes pattern recognition is not built in.

5 AGENT HUMAN COMPARISON

Overall, I believe the agent and human are very similar in terms of the approach. When I look at any of the problem sets, the first thing I do is try to understand and learn any sort of relationship between all the images. Naturally, I look at relationships at a row level, then column and then a diagonal relationship. Although the RPM agent did not require a column relationship section, but I had considered to build that if my agent was performing poorly. Furthermore, I would also try to identify visually the patterns/relationship:

- If there were pixels being added or subtracted
- If two images were a bit-wise operation of each other to produce the third
- If the images were related to each other via a rotation or mirroring
- If the shapes were filled in or not
- If there was a pattern identified with shapes
- If the shapes were moving inwards or outward

The above are merely a few aspects of the images I would look at and my RPM

agent does look at these as well. Provided more time, I could have attempted to add in more logic to make the RPM agent smarter and populate more information in it's knowledge representation.

A human would also first learn about these relationships, generate a possible solution in their mind, then attempt to find an answer which fits best to the learnt knowledge about the images and test them against the generated solution they have in their mind. In some cases, I also attempted to rule out several possible answers which would not make sense, and then if there are two very similar answers I would attempt another round of comparison and checks to see which answer best fits the relationships I have learnt from the supplied images (A, B, C etc). This is the same approach as the agent of using problem reduction and voting to find an answer when there are multiple possible answers.

The key differences between humans and the agent would be speed and the ability to compute pixels. The agent is clearly much faster than human. Since the agent cannot 'visualize' the images, the agent needs to compute individual pixels - whereas I just have to see the image and I can detect a pattern without needing to actually calculate how many pixels and pixel changes occur in the RPM problem.

Lastly, I found that us human are able to quickly figure out relationships that are bit-wise operations and exact matches but the agent may run into troubles with it and therefore requires a threshold level to allow some degree of error. This is the difference of being able to 'visualize' versus having to compare pixel by pixel. This definitely makes me think if the agent was provided a camera to detect the images and supplied with similar logic approach identified in this paper - then would it outperform humans now that it can 'see'?

6 APPENDIX

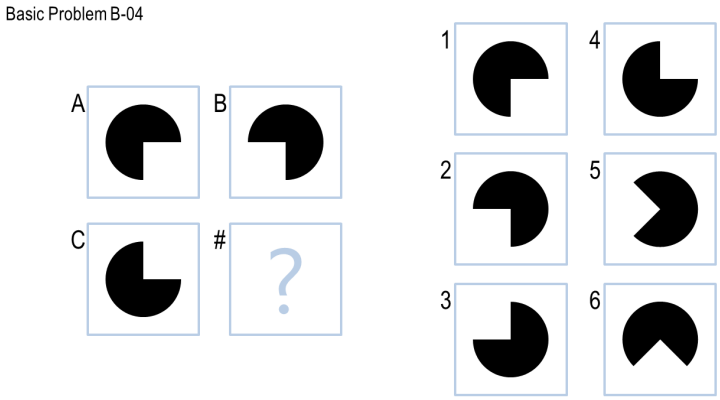

Figure 2—Example of 2x2 matrix with rotated images

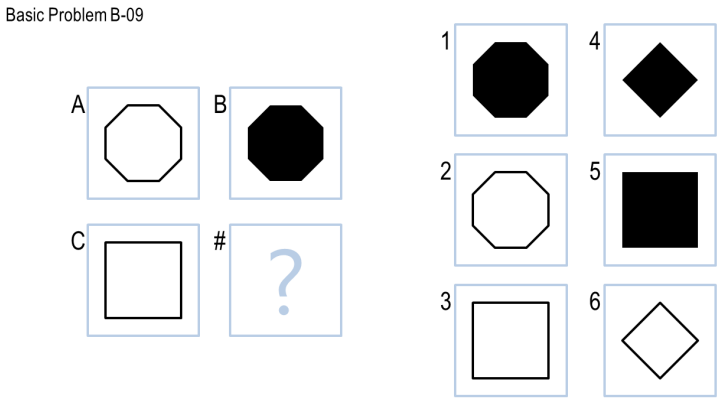

Figure 3—Example of 2x2 matrix shape detection and pixel change

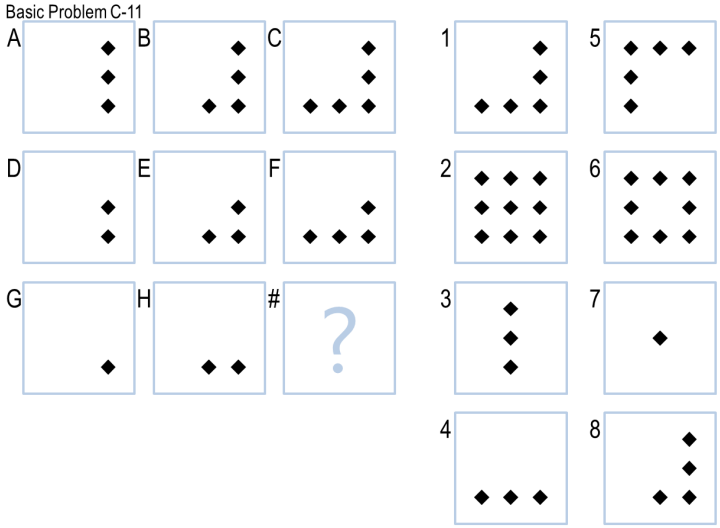

Figure 4—Identifying center location of dark pixels

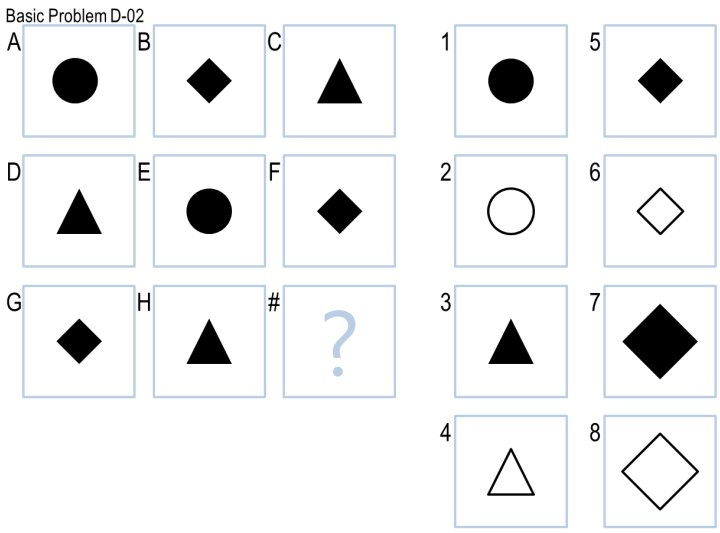

Figure 5—Diagonal relationship

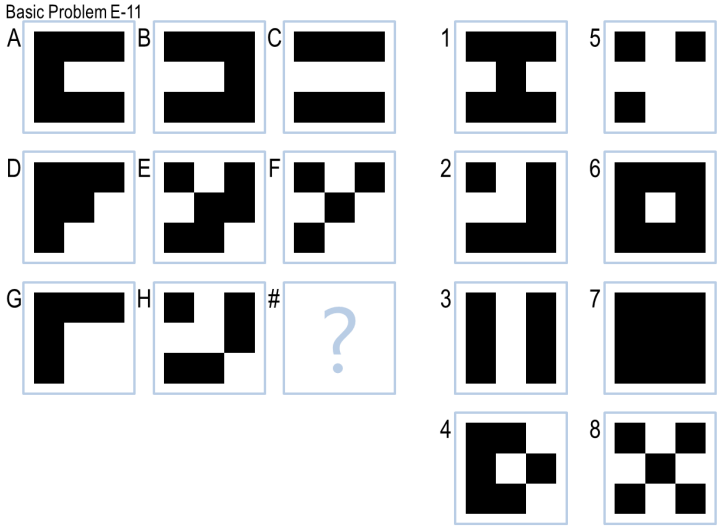

Figure 6—Using bit-wise operator OR

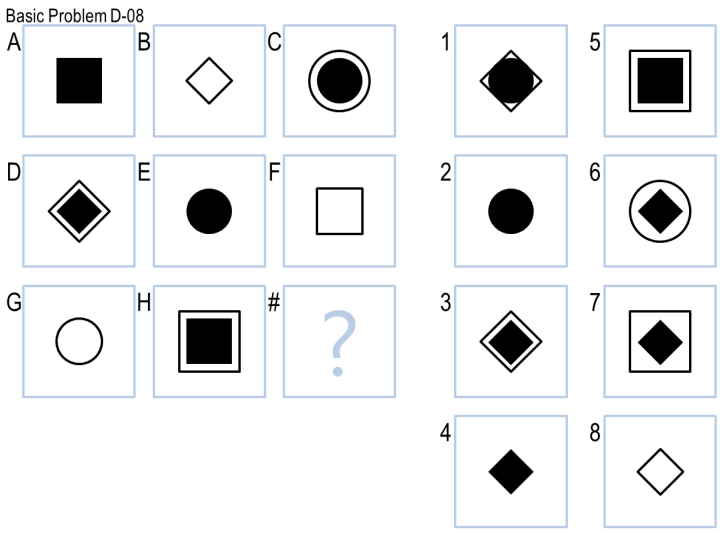

Figure 7—Struggles with shape detection

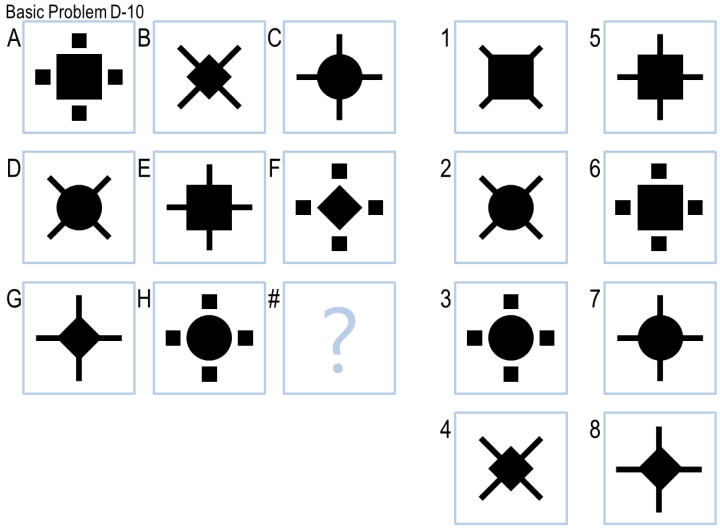

Figure 8—Struggles with identifying patterns

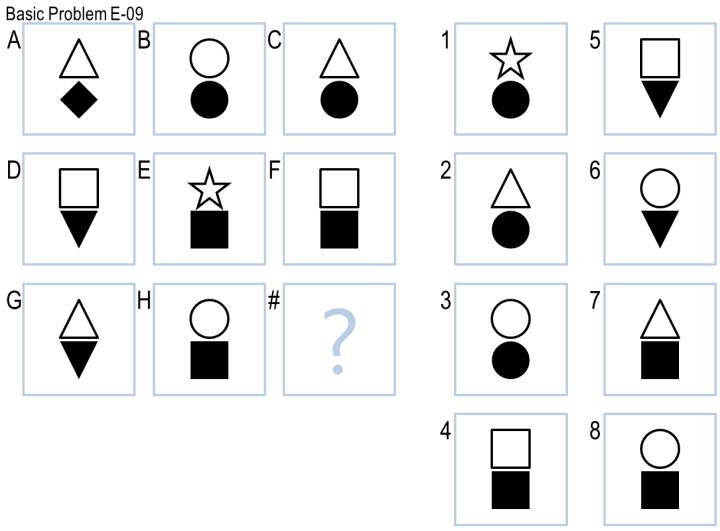

Figure 9—Struggles with multiple patterns in a row