PROJECT-2 REFLECTION [Vinodh Kumar Sunkara, Yahoo Inc.]

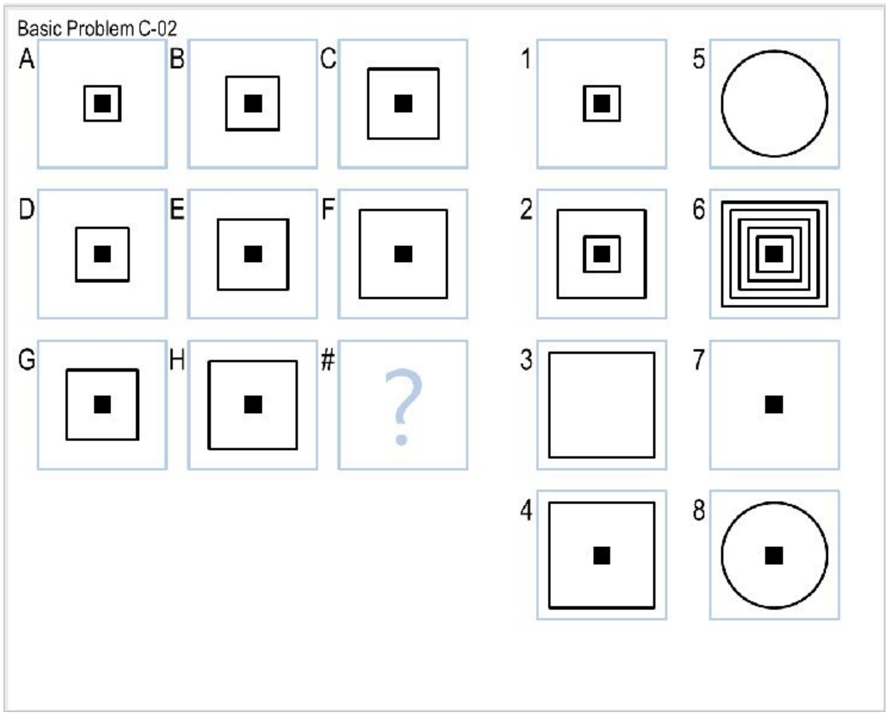
Introduction

Raven’s Progressive matrices have been used to test the intelligence of AI agents for a long time. Implementation of AI agent should definitely replicate human cognition while solving Raven’s Progressive matrices. It’s important to have an insight on how human solves each problem before training the AI agent to solve it. Humans apply different methods to recognize shapes and patterns in the problem. Some methods can lead to answer very confidently and a few others need options to pick the right choice that fits in pattern. There are some human reasonings that can lead to false positives too if the reasoning is not very tight. Analogous to this, AI agent also acquires a set of rules that are applied intelligently to reduce false positives. The less confident rules are applied after all the confident rules are explored on a given problem.

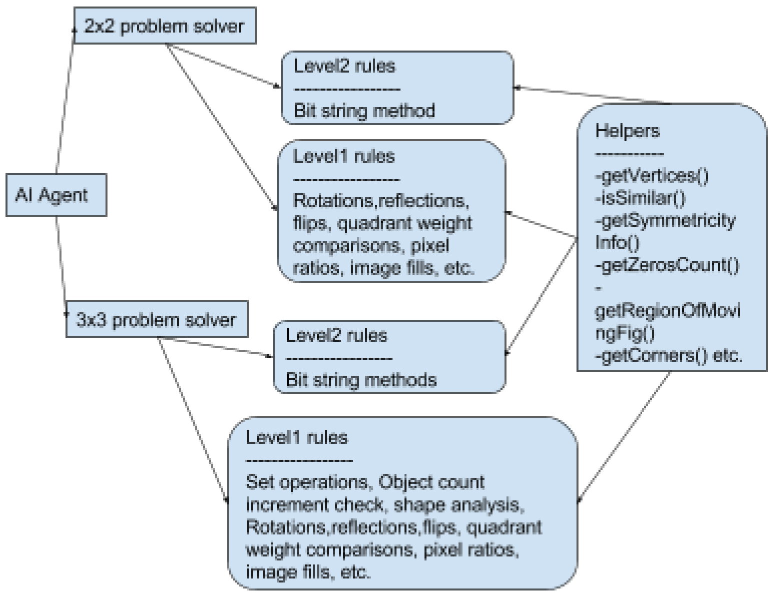
AI Agent construction

Agent has 2x2 and 3x3 problem solvers. Each problem solver has two levels of rules. Level1 rules are applied horizontally and vertically to decide upon an answer. Horizontal application is to observe transformation undergone to produce C, F in A-B-C and D-E-F sets respectively and produce answer image ‘#’ using same transformation on G,H images. Vertical application of rules realizes patterns across A-D-G, B-E-H sets and apply on C,F to generate the answer image ‘#’. Level2 rules derive a common pattern considering the whole image set A-B-C-D-E-F-G-H at once and derive the answer that best completes the pattern. I designed the rule set that’s applicable both horizontally and vertically by passing a direction parameter to deal with right image ordering.

An example of a level1 rules application is in below problem:

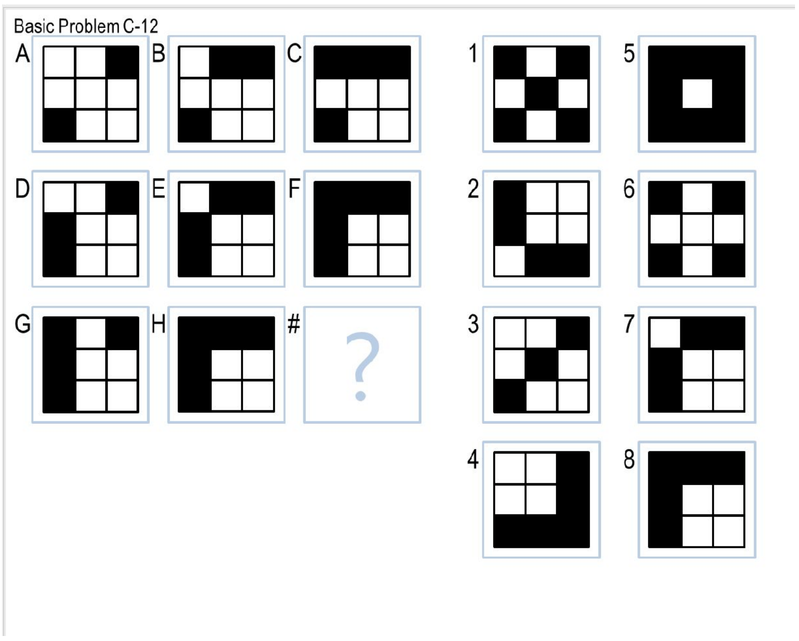


The level1 rule that can observe the pattern of outer square expansion in A-B-C, D-E-F sets can be used in same way on vertical sets A-D-G and B-E-H. In this case, the rule agrees upon same answer in both horizontal and vertical direction. In some other problems, it can generate an answer in one of the two directions and some other rule is applied in the other direction to end up on common answer. My solvers finalize an answer only if both the horizontal and vertical direction ends up in same answer through any level1 rule applied individually. Since, level2 takes all images at once into consideration, it's ordered high in priority. Level1 rules are touched by an agent only if level2 can't come up with a solution. Even within the level1 rules, there is a priority ordering where most confident rules are first to be applied.



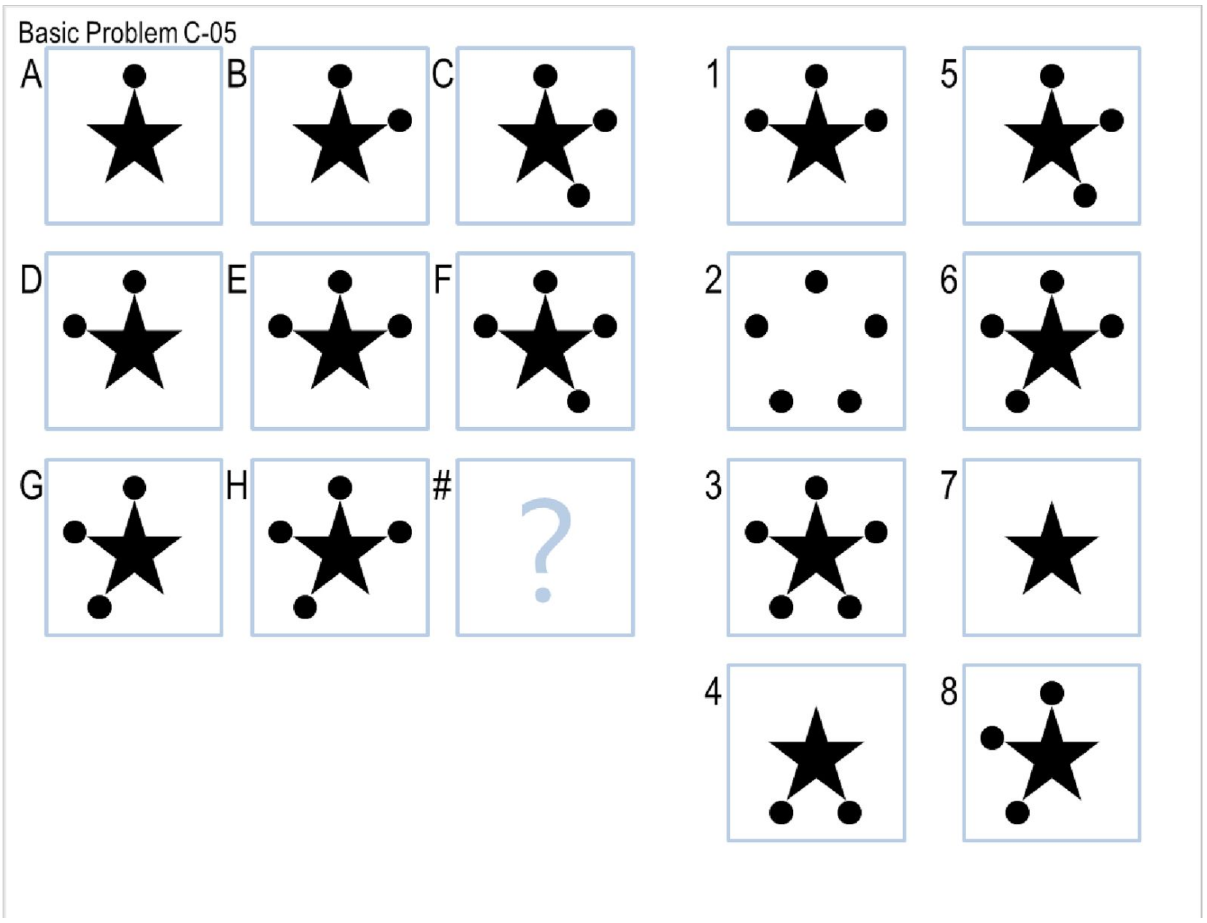
Methodologies

An example of a level2 rule used is for the below problem. The bit strings are generated for all figures and the missing bitstring is derived from the bit patterns. The corresponding image for the generated bitstring is picked from options as the answer.



We can very well extend the same rule in future to solve similar problems with different analysis on bit strings. A portion of bit string is unchanged in all figures and it's borrowsed to answer bitstring while rest of it derived using different techniques like maximum number of 1bit or 0bit, and other bit manipulations.

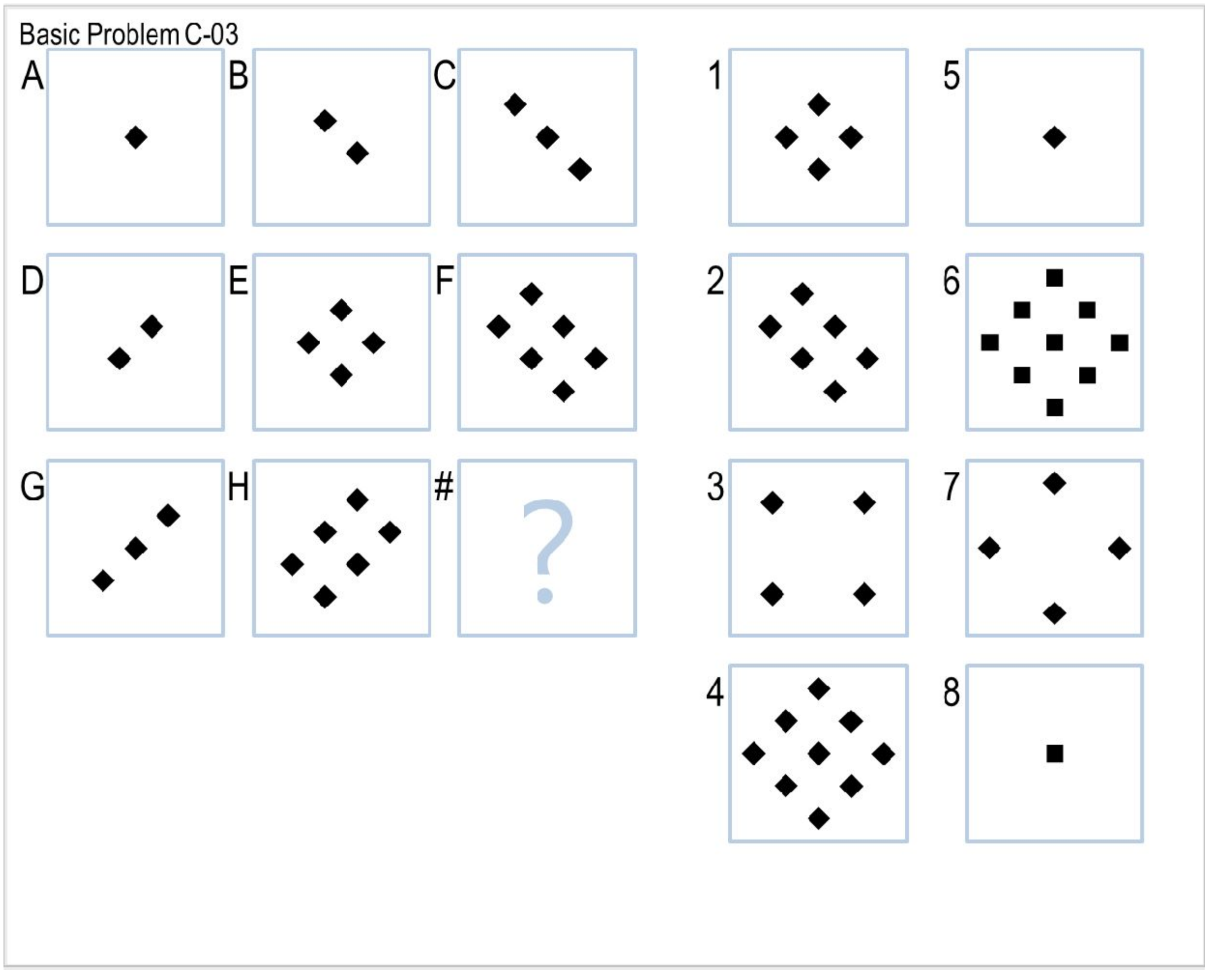
Level1 rules also explore set operations like in this problem:



This problem is not exactly solved by spotting the circular portions and their addition horizontally or vertically in patterns. It's solved differently using multiplication of images and nonzero pixel count comparisons.
A*B=B, B*C=C, D*E=E, E*F=F, G*H=H and same pattern is applied on different answer options to decide upon the answer. Here the tester is less dumb in Generate-and-Test method.

There are two answer options 3,6 that respect the pattern extension H*(#) = (#). We also use the nonzero pixel count decrease by as much as the circular portion weight to denote addition of a new dark circle and decide on option 3.

The above problem has to deal with the arrangement of circles around, but a much simpler version is shown below to denote the pattern of increasing objects. Below rule is put top in priority among level1 rules and applied before the above rule.



A simple zero pixel count pattern check solves the above problem. It's considered a confident rule since the count change is mostly accurate with an assumed little error rate. Agent applies this rule in middle order on any problem after the very simple equal figure comparisons, etc. rules. This method will land up in two options 4 and 6, after applying assumed error rate in comparisons. We need a method to filter between 4 and 6. I've a helper isRhombus() on the image to check if it's a rhombus or square and that decides option4 as the answer.

Helpers

There is a big set of helper methods I designed that are used in both 2x2 and 3x3 solvers. A few helper methods are described below:

- getVertices: To get vertices count
- isSimilar: Takes a percentage parameter to compare image with assumed error rate
- getSymmetricyInfo: Returns an object with keys describing horizontal and vertical symmetry and also if horizontal and vertical dividers for image has any dark spots
- isRhombus: This method says if the figure is a rhombus
- getTopLetCorner: returns the top left corner of the image
- getBottomRightCorner: returns bottom right corner of the image
- getRegionOfMovingFigure: This method is used to enumerate and give a region number if figure is moving from top left to bottom left along the boundary.
- getDiagonalShadeType: Gives an object that describes if the diagonal exists and it's from top left to bottom right or top right to bottom left aligned.
- getRotatedImage: rotation with customized angles
- getReflectedImage: reflection across customzied axis
- getFlippedImage: Flip images along different axis etc.

Performance

My AI agent is highly performant mainly because of the intelligent ordering of rule set. It takes on an average around 45ms per RPM. It needs this time to process different rules in horizontal and vertical directions to come up with an accurate answer. My agent currently solves 100% basic problems in sets B and C. The scores are as follows: 12/12 basic B, 12/12 basic C, 5 correct in C ravens, 5 correct in C test, 6 correct in B test, 4 correct in B ravens. The known issues that might cause a few errors from test set are described in below section.

Risks

Currently, my AI agent applies all rules horizontally on A-B-C and D-E-F sets and also vertically on A-D-G, B-E-H sets. It decides on answer only if horizontal and vertical options selected match. I've made it this way for higher accuracy. I can very well have a final fall back to go with an option decider either by horizontal or vertical pattern to get few more problems correct in Test. I would rather want to make my agent stronger with current approach to get 100% test problems right with higher accuracy. However, basic problems have already proved the accuracy with 100% correct answers. There is also an error rate I assumed on figure comparisons. While figure generation or initial figure comes up with a 0.5% error, after manipulations like multiplication, addition, reflection, etc. on image increases the error rate. While assuming this error rate to as high as 3 percent in few cases, some false positives can occur. The tight bound of horizontal and vertical rule agreement can eradicate this to the most extent.

Conclusion

The whole AI agent architecture and functioning replicates human cognition and it does function as expected by me to the most extent. But this doesn't make me happy! The issue is that I feel it's more a limited replica of myself than an artificial agent on it's own. All the knowledge it has is installed by me which is yet limited. Once it encounters a brand new problem, it can't learn or do anything on it's own except to flunk. I would like to add the ability to self learn on my agent that it learns by itself on any new problems by analyzing individual images and realizing the patterns. Hoping to achieve this to a good extent by end of this course.