AI Agent for IQ test

[Arti Chauhan: Nov 24 2016]

Introduction

This paper describes design and implementation of an AI agent that can solve 2x2 as well as 3x3 Raven's Progressive Matrices (RPM).Being new to image processing and python, I initially found this project intimidating and at times frustrating, but in long run, it helped me learn few things about my own cognition and build upon it.

Some key aspects of Agent's design such reasoning method, answer selection strategy, agent's overall architecture hasn't fundamentally changed since P2. For sake of completion, I have retained high level summary of that in this paper.

Problem-solving approach, design considerations and payoffs *(Q1)*

Agent uses **'Classification'** to categorize RPM problems either as 2x2 or 3x3 and then invokes appropriate plan (Fig-2) to solve that problem. Each **Plan** has a rule-set that helps agent determines sequence of transformations to execute to achieve the goal. Agent uses **Frames** for knowledge representation and **Generate and Test** in conjunction with planning for problem solving. With help of Pillow (Python Image Processing) library, Agent employs **Visuospatial Analogy** at its core to solve the problem.
Below is simplified view of Agent's framework.

Step-1: *Agent reads visual input and stores the information in appropriate data structure (RPMFrame Class).*

Step-2: *Agent is equipped with problem classification criteria and a plan for each category (2x2 or 3x3). Agent loads transformation rules from rule set for each plan. Rules are defined in order of complexity, ie simpler transformations take precedence over complex ones. E.g. Agents tests for simple row/column wise 'AND', 'OR' or 'XOR' before checking for cyclic diagonal pattern, reflection before rotation, and rotation before 'object addition/deletion' etc.*

Step-3: *In step-3, Agent starts applying given rules sequentially to test horizontal, vertical and diagonal transformation.*

- *For a 2x2 problem, it applies the given transformation-rule to Frame A and compares transformed image against Frame-B per criteria described below. If comparison result is within acceptable threshold (i.e. two images are determined to be similar), it applies same transformation-rule to Frame-C and compares it against each answer choices. Process is repeated for column wise comparison (AC -> B?)*

  Criteria for image comparison: *To determine how different two images are, Agent computes diff_score which is ratio of number of pixels that differ in both images divided by mean pixel count [1]. If diff_score is less than DIFF_THRESHOLD (2), two images are considered to be similar.*

- *For a 3x3 problem, approach is similar to 2x2 except that combination of images compared differ. These combinations are listed in table below. In P3, Agent learnt few more relationships (listed under 'OTHER').*

2x2

A	B
C	?

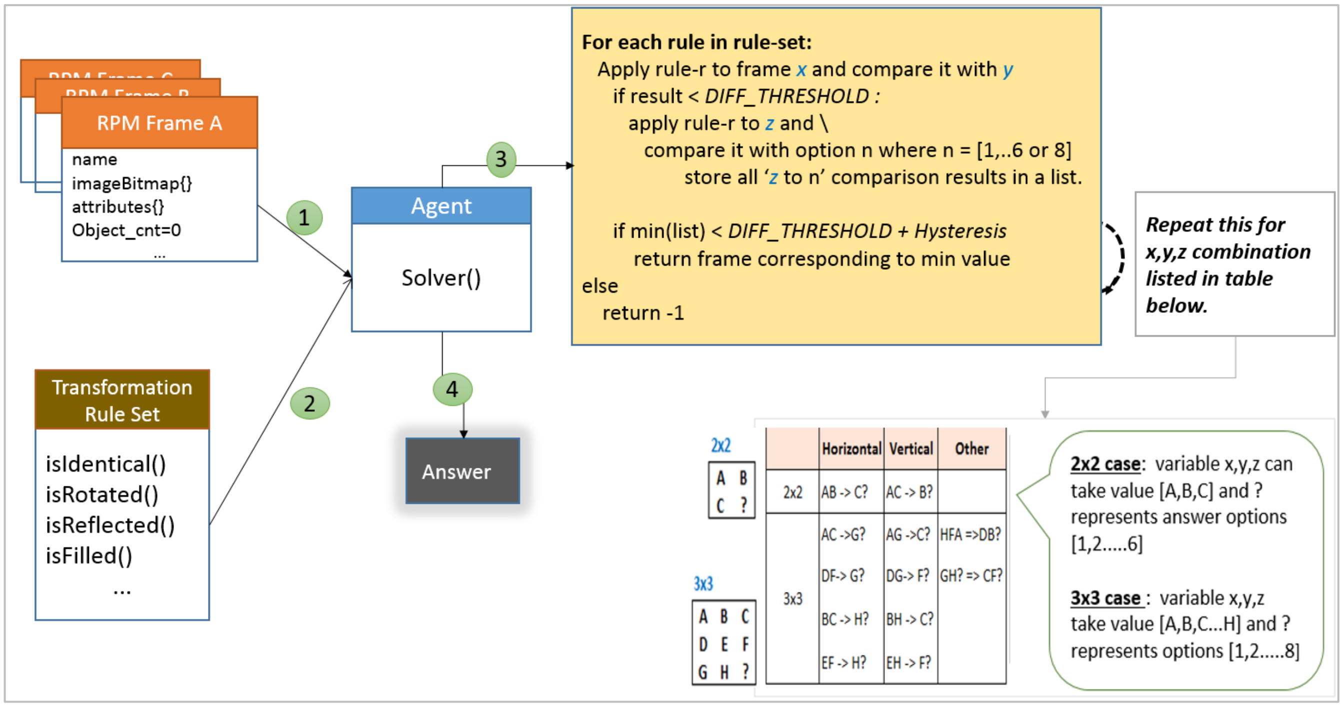
3x3

A	B	C
D	E	F
G	H	?

	Horizontal	Vertical	Other
2x2	AB -> C?	AC -> B?	
3x3	AC ->G?	AG ->C?	HFA =>DB?
	DF-> G?	DG-> F?	GH? => CF?
	BC -> H?	BH -> C?	
	EF -> H?	EH -> F?	

Step-4: *Diff_score is computed for all C->N answer choices and stored an array. If minimum value of C->N comparisons is below a certain threshold, frame corresponding to minimum value is returned by Agent as answer, else -1 is returned, which in turn triggers next rule in the set.*

- *This process continues until all rules are tested. If there is no answer found that satisfies threshold criteria, Agent returns -1(problem is skipped) and moves to next problem.*



RPM Frame C
RPM Frame B
RPM Frame A
name
imageBitmap{}
attributes{}
Object_cnt=0
...
1
2
3
4
Agent
Solver()
Answer
Transformation Rule Set
isIdentical()
isRotated()
isReflected()
isFilled()
...
Repeat this for x,y,z combination listed in table below.
2x2
A B
C ?
3x3
A B C
D E F
G H ?
2x2 case: variable x,y,z can take value [A,B,C] and ? represents answer options [1,2.....6]
3x3 case : variable x,y,z take value [A,B,C...H] and ? represents options [1,2.....8]

```
For each rule in rule-set:
  Apply rule-r to frame x and compare it with y
    if result < DIFF_THRESHOLD :
      apply rule-r to z and \
        compare it with option n where n = [1,..6 or 8]
            store all 'z to n' comparison results in a list.

    if min(list) < DIFF_THRESHOLD + Hysteresis
      return frame corresponding to min value
else
  return -1
```

	Horizontal	Vertical	Other
2x2	AB -> C?	AC -> B?	
3x3	AC ->G?	AG ->C?	HFA =>DB?
	DF-> G?	DG-> F?	GH? => CF?
	BC -> H?	BH -> C?	
	EF -> H?	EH -> F?	

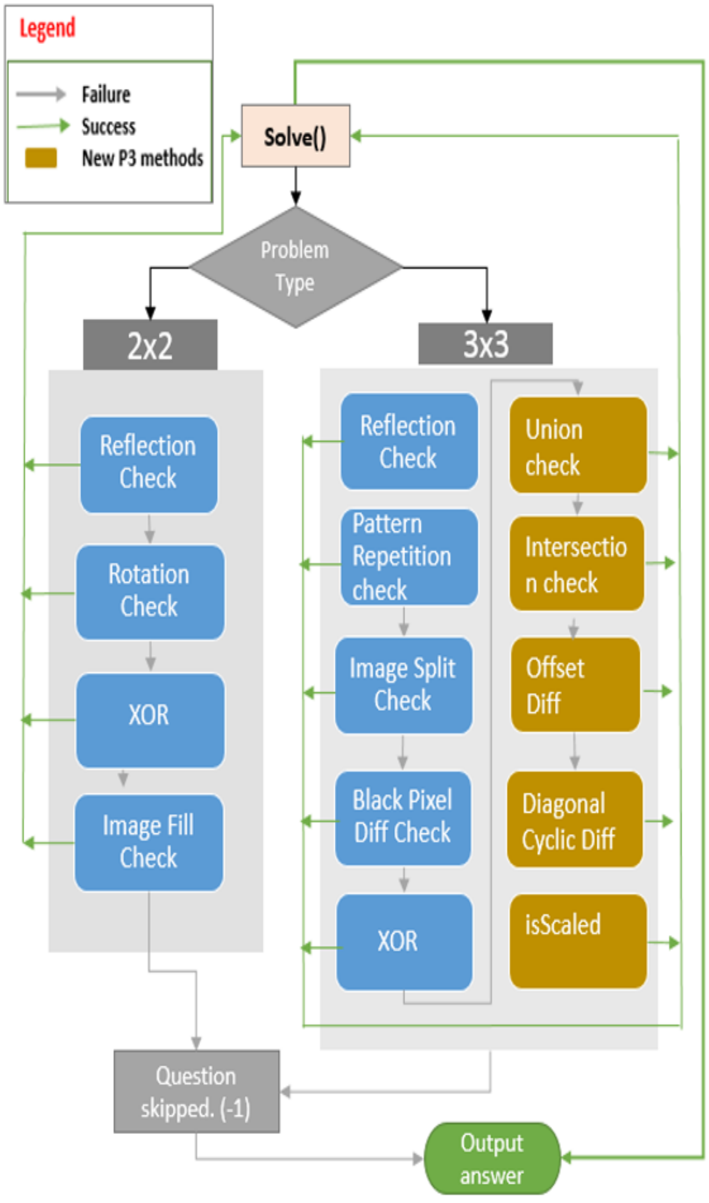
Fig-1: Agent's Framework

Design considerations: From the very beginning I have strived to keep my agent's design simple and modular, which allowed

- me to handle any type of input (propositional or visual) and helped me move easily from verbal to visual approach in P1.
- increased code reuse factor – For 3x3 problems, I was able solve 50% of Basic-C problems by reusing transformation rules from P1 and 80% of Basic D& E with just 4 transformation rules.

Answer selection *(Q2)*

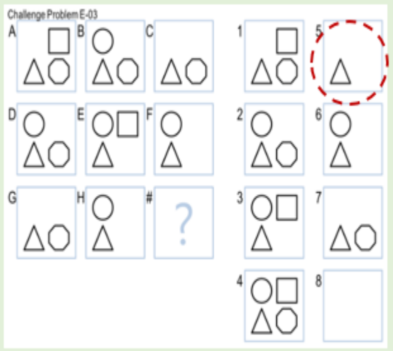
Agent uses the same answer selection strategy as used in P2 except that in P3, Agent's library was enhanced to recognize more complex relationships. Below is detailed explanation of answer selection process.

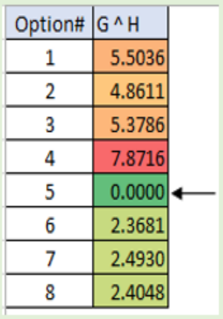


- Agent goes through series of transformations in the order shown on the left.
- Each 'transformation-check' module compares transformed image with option image and stores the difference (non-similarity) score in an array.
- If minimum value in that array is below a certain threshold then index of that minimum value is returned as answer. Else -1 is returned.

Example below shows answer selection for Challenge E-3 problem. Agent starts with Reflection check and goes through series of transformation checks until it finds an answer that satisfies confidence threshold. In this case, when Agent arrives on '**Intersection Check**', following values are yielded.

1. Diff_score(A ^ B ,C) = 0.0034
2. Diff_score(D ^ E ,F) = 0.0045
3. Agent proceeds with step 4 as result from #1 and #2 are below threshold (2).
4. Agent computes difference between (G ^ H) and all option images and stores results in an array.
   - Diff_score(G ^ H , *N*) = [5.5036, 4.8611, 5.3786, 7.8716, **0.0001**, 2.3681, 2.4930, 2.4048]
   - min_value = 0.0001 , min_index = 5
5. Since min_value < threshold , Agent returns 5 as answer.





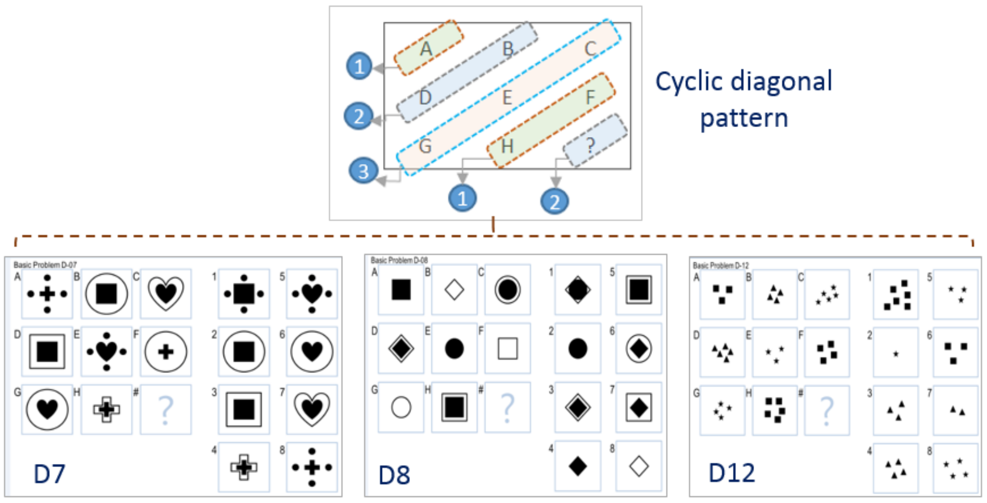
Option#	G ^ H
1	5.5036
2	4.8611
3	5.3786
4	7.8716
5	0.0000
6	2.3681
7	2.4930
8	2.4048

Fig-2: Answer selection process

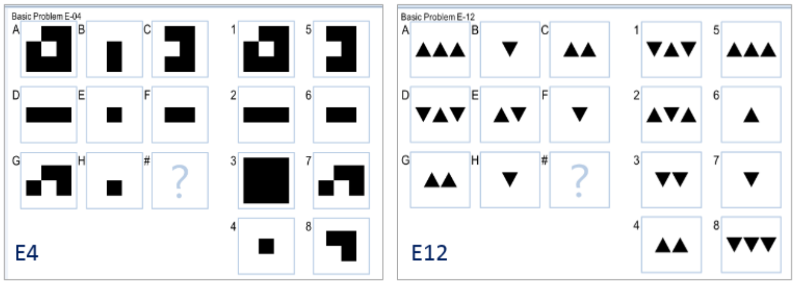
Challenges/ learning in P3:

- Project-3 was interesting in the sense that I ended up spending 80% of my time on 20% of the problems. I couldn't help but think if I am witnessing Pareto Principle here.
  - While I was able to solve 19/24 problems with relative ease using 3 generic methods (AND/OR/XOR), few problems took me longer than expected. D7, D8 and D12 were my favorites as they gave me most heart-burn. While Agent could see all three problems exhibit similar structural correspondence between AHF =>GEC=>DB? (Which I called 'Cyclic Diagonal', for lack of better term) but transformation required to achieve this relationship varied for each problem. Agent had a hard time coming up with one generic method for these 3 problems.

  - This raises another point – how to balance generality vs. specificity in an Agent. Eg: Agent was able to solve 9/12 problems in Set-D with just one method described in Figure-3.But for that I had loosen DIFF_THRESHOLD to make it generic enough. This reduced processing time but costed me 2 test questions. With reversal of DIFF_THRESHOLD to its original (P2) value, I had to incorporated 5 more new transformation-checks (increased specificity), which helped me recover lost point on test (5% gain in accuracy) but increased response time by 57sec (35%). Is this a good tradeoff? May be not for large problem sets. At this point I don't have a real good answer to how can an Agent strike a balance between accuracy vs. efficiency?



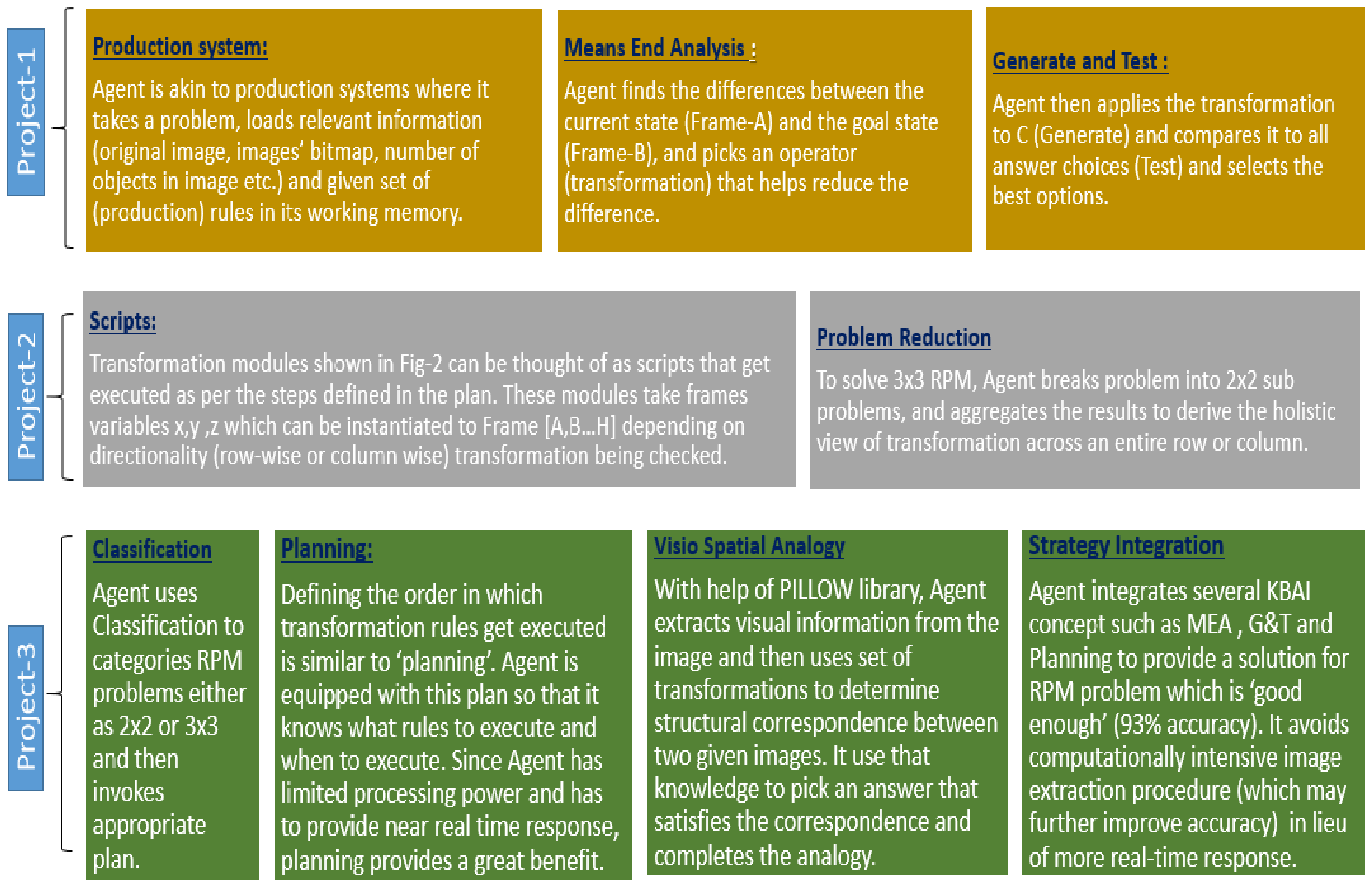
- Throughout the project, Agent has been learning incrementally. Eg: When Agent saw E4, it devised an approach where A XOR B (after offsetting images to zero on x axis) yields C. Agent deployed same approach for E12 but failed. Here, it had to VERTICAL_FLIP B before applying XOR. There can be many more transformations that Agent hasn't seen such as B is scaled or is at an offset or flipped and/or unshaded. I struggled with to what degree one should train his agent to tackle unseen (yet probable) cases because such cases can be infinite. I added few more transformation rules I could think of but it came with a price – it increased Agent's computational time, hence impacting responsiveness. This hits home with dilemma Prof Goel described how an AI Agent with limited resources, bounded rationality, near real-time requirement can tackle all the problems in a dynamic world.



- Another challenging aspect of this project was how to incorporate advanced learning concepts in Agent. As described in example above, Agent learns incrementally - it starts with a specific model 'A XOR B =C 'for E4, then enhances that rule to include A XOR Flip (B) =C when it sees E12. Given limited number of questions Agent is exposed to, it is bound to have gap in its knowledge. In any learning system, learner is provided feedback that a particular response/solution is incorrect, which triggers corrective action in learner. But here when Agent fails a Test question, it doesn't get any feedback, hence can't learn why it failed and consequently, would make same mistake again. I personally found it challenging to apply any of the advanced concept especially learning/Meta reasoning to my Agent.

Relation to KBAI methods *(Q4)*

As the course progressed, I could connect my Agent's design to several KBAI concepts taught in the class. Below I have listed connections I could make at each stage of the project.



Project-1

Production system:
Agent is akin to production systems where it takes a problem, loads relevant information (original image, images' bitmap, number of objects in image etc.) and given set of (production) rules in its working memory.

Means End Analysis :
Agent finds the differences between the current state (Frame-A) and the goal state (Frame-B), and picks an operator (transformation) that helps reduce the difference.

Generate and Test :
Agent then applies the transformation to C (Generate) and compares it to all answer choices (Test) and selects the best options.

Project-2

Scripts:
Transformation modules shown in Fig-2 can be thought of as scripts that get executed as per the steps defined in the plan. These modules take frames variables x,y ,z which can be instantiated to Frame [A,B…H] depending on directionality (row-wise or column wise) transformation being checked.

Problem Reduction
To solve 3x3 RPM, Agent breaks problem into 2x2 sub problems, and aggregates the results to derive the holistic view of transformation across an entire row or column.

Project-3

Classification
Agent uses Classification to categories RPM problems either as 2x2 or 3x3 and then invokes appropriate plan.

Planning:
Defining the order in which transformation rules get executed is similar to 'planning'. Agent is equipped with this plan so that it knows what rules to execute and when to execute. Since Agent has limited processing power and has to provide near real time response, planning provides a great benefit.

Visio Spatial Analogy
With help of PILLOW library, Agent extracts visual information from the image and then uses set of transformations to determine structural correspondence between two given images. It use that knowledge to pick an answer that satisfies the correspondence and completes the analogy.

Strategy Integration
Agent integrates several KBAI concept such as MEA , G&T and Planning to provide a solution for RPM problem which is 'good enough' (93% accuracy). It avoids computationally intensive image extraction procedure (which may further improve accuracy) in lieu of more real-time response.

Shortcomings and improvements *(Q3, 5)*

Below I have listed some of the shortcomings and plan to remediate it, which will help make agent more generic and efficient.

- Image Analysis - In P3, Agent was enhanced with Connected Component logic but it fails when images are overlapping. Moreover, it increased Agent's processing time noticeably with minimal gain in accuracy. Agent is working at highest abstraction level where it is agnostic to object's, thus may fail to handle more complex problems.

- Threshold: In each phase of the project, deriving a Threshold value (used to determine similarity between two frames or transformations) that balances generality and specificity of the Agent was the most time consuming part. Though I tried to derive this threshold in a scientific way by computing the PDF of diff_score for all basic questions, this approach was limited by questions Agent has been exposed to. Moreover, in several cases I had to tweak the value to maximize test-score, which dings Agent's generality.

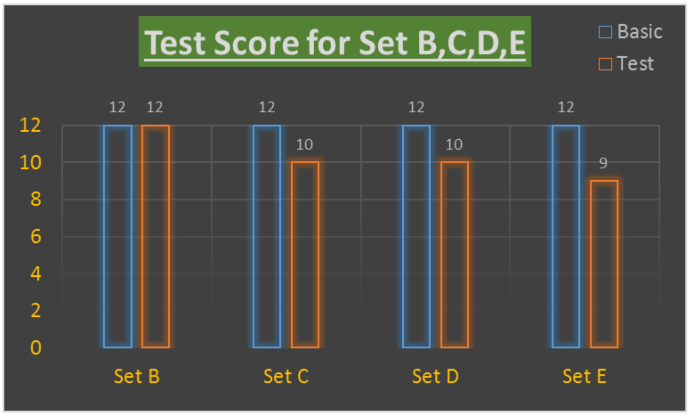
- Memory: Agent doesn't have a long term memory .If it makes a mistake for a given problem, it will make that mistake again when same problem is encountered. This is not a fundamental issue with the design but rather a question of evolution of Agent to incorporate learning and self-correction.

- Hybrid Approach – Since P1, Agent is fully relying on visual approach. I don't see it as a shortcoming as increased complexity didn't justify the minimal accuracy gain with hybrid approach. Nevertheless, Agent's design has appropriate hooks built-in to extract attributes using propositional logic, if need be.

- Rotation generality: One of the rules related to rotation is not generic enough. Agent is currently built to handle rotation in increments of 45 degrees, which is a limitation and can obviously fail.

Performance *(Q6)*

a) Given only handful of transformation-rules in Agent's knowledge base, it was able to score 89/96 (93%) on Basic and Test set.

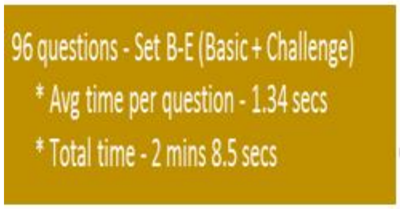


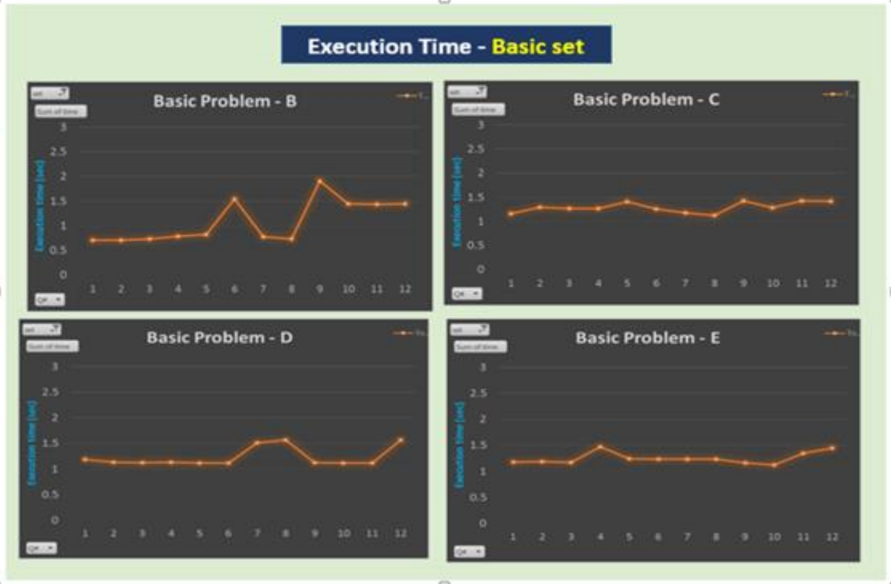
b) Generality: Agent is generic enough to solve 85% of 3x3 set with few transformation-checks. With addition of just four new transformation-checks in P3, Agent was able to solve 19/24 questions and tackled unseen Test problems with decent accuracy.
Eg: Agent was able to solve 8/12 questions in Set D with just one transformation check, which worked as follows.

```
If variance in row_1 and row_2 are similar then variance in row_3 is expected to be same as row_1 and 2.
        var_row1 = diff(A,B) U diff(A,C)
        var_row2 = diff(D,E) U diff(D,F)
        if abs(var_row1 - var_row2) < threshold(2):
            var_row3 = diff(G,H) U diff(G, i )
                     where i is the option image that produces var_row3 row_3 similar to var_row1 and row2.
                     diff is ImageChops.difference
```

Figure-3

c) Execution Time: Agent is able to finish Basic and Challenge questions (96 questions in all) in 2mins 9secs. So efficiency isn't a concern at this stage but again problem space is quite small.


96 questions - Set B-E (Basic + Challenge)
* Avg time per question - 1.34 secs
* Total time - 2 mins 8.5 secs



Time Complexity:

Agent's Time complexity is O (N * T * x)
where
N = number of RPM problems
T = Number of transformation-checks in rule set
x = Number of answer choices (6 for 2x2 and 8 for 3x3)

Since x is a constant (a small value), its impact will diminish as N and T approach infinity and hence can be ignored, giving time complexity as O (N*T). One way to reduce time-complexity is by reducing/optimizing T. This can be done by

1. *Minimizing the number of transformations performed*
2. *Minimizing the similarity comparisons performed within each transformation.*
3. *Optimizing similarity-check function.*

Throughout the project, Agent strived to improve the response time and efficiency. It was achieved by

1. Employing a confidence threshold, which minimizes 1 and 2 mentioned above.
   - If diff_score of two given frames exceeds a certain threshold, Agent abandons that transformation check. Eg: while checking AB->C? , if image delta between transformed-A and B is above certain threshold (ie images are not similar), Agent returns and won't check C:?, avoiding 6 extra comparisons.

2. However, as project progressed from P1 to P3, increase in transformation-checks was unavoidable. To offset this, in P-3, Agent was enhanced with an approach to prune the search space. This was done by Depth First Search Connected Components algorithm, which helped Agent determine number of objects in an image. Agent used that information to rule out options that didn't fit the object count pattern in the given problem set. But this enhancement came with a cost. Agent's response time increased almost two folds, with very little gain in accuracy. Though I implemented this logic, I decided not to use it as cost to benefit ratio was high.

Time complexity - Best & Worst case

		2x2	3x3
Num of Transformation checks in rule set	From Fig-2	4	5
	accounting for row-wise and column-wise comparison,	8	10
Function calls to transformation modules	best case	1	1
	worst	8	10
Number of transformations to check (leading to image comparison)	best case	7	9
	worst	46	180

For 1 RPM problem
Best case : 1st transformation gives the result.
= 1 (AB comparison) + 6 (C? comparison) =7

Worst case : Agent goes thru all transformation mdoules (8 for 2x2)
= (1+6) * 8 = 48

To generalize : if T= num of transformations to check and X = number of answer choices, then num of comparisions
best case = (X+1)
worst case = (X+1) * T for 2x2
= 2((X+1)*T) for 3x3
In best case , agent will take constant processing time O(1).
Worst case => O(T) .

For N RPM problems
To solve N problems, agent will take O (N) in best case and O(N*T) in worst case . This assumes X stays constant(6 or 8) and hence have negiligible impact as N and T approach infinity

Reasoning Method *(Q7)*

Agent is extracting visual representation to reason over images. Agent mostly relies on PILLOW APIs for this purpose but in few cases those APIs were not sufficient to detect the transformation reliably. In such scenarios, Agent converts the image to a numpy array and perform some thresholding before checking for any transformation.

Cognitive Connection *(Q8)*

Over the course of the project, my Agent has progressed from 'crawling' to 'walking' but has a long way to go before it can 'run'. It exhibits some aspects of human intelligence where it reasons and performs some learning but lacks meta-thinking.

It mimics human's visuospatial reasoning approach when solving RPM problem. Most people solve RPM problem by studying the top row, incrementally generating hypotheses about how the objects varied across that row, and then looking at bottom row to test those hypotheses. Agent mimics this behavior - It solves the problem by determining the variance in top row and then selecting the answer that best fits for bottom row. If no answer scores within acceptable range it looks down columns.
It goes for easier transformation first and if no answer is found, it then tries for more complex transformations, just like we humans do.

It differs from human as it has extremely limited meta-cognition and self-learning ability. It is not able to determine why and where it failed and employ self-correction. For that, it needs to be either exposed to a large number of problems to fill its knowledge gap or given feedback when it fails or succeeds if training set is very small, both of which were missing in this project.
Faintest relation Agent shows to 'thinking' is in the sense that when row-wise transformation-check fails, Agent 'thinks' about what went wrong and try's to correct it by modifying the approach (perform column-wise and or diagonal check). If that still doesn't give desired result, it goes for other transformation-checks it has in its working memory.

Humans are deft at adapting old solution to new problems. But my Agent's adaptation capability is limited. It can adapt an old solution only to a certain degree. That's partly because it lacks deeper understanding of the image. Eg: Agent has no problem detecting objects when they are not overlapping but fails when they do. It can't recognize shapes like star or heart.

All in all, this course was an excellent journey to explore my own cognition and understanding the complexities involved in developing artificial intelligence.

References

1. *https://rosettacode.org/wiki/Percentage_difference_between_images*
2. *Udacity video Lectures – KBAI*