Brad Ware
bware8@gatech.edu
Assignment 2

Using Semantic Networks to solve Raven's progressive matrices

When trying to solve Raven's progressive matrices the most complex task is relating the different images to each other through similarities and differences. Each paired image represents a "transformation" of its partner, and that is the task of an intelligent system – recognizing these patterns and applying them to another image. From a high level perspective, semantic networks simply represent relationships between objects in a system. These relationships are usually based on a desired-goal, an adhered structure, and semantic knowledge about each individual object. Visually, these relationships are just simple links connecting nodes based on these 3 factors and even more, depending on the problem statement.

The most challenging side of semantic networks is to understand the problem so that one can adequately convey relationships between separate figures that will reach the goal. In this case, we are working with images that contain different objects that resemble shapes of all sizes, types, and locations on the frame. In the case of Raven's progressive matrices, we know that each image will contain some variation of another, so we can base our relationships off of that assumption. So to start solving this problem, the semantic networks need to be defined in some way that matches the problem. Each object in an image is important, because they represent a relationship that could weigh choosing its matching pair. Therefore, as a pre-processing step, each object in an image needs to be given values based on it's type, location, and size. These can be stored in an external object that encapsulates these characteristics. From there, we now know how to properly identify each image, but still have no way of relating them. But we are given an example pair of images that will allow us to derive our answer. It will be easiest to represent this through an example, which can be found on the next page:

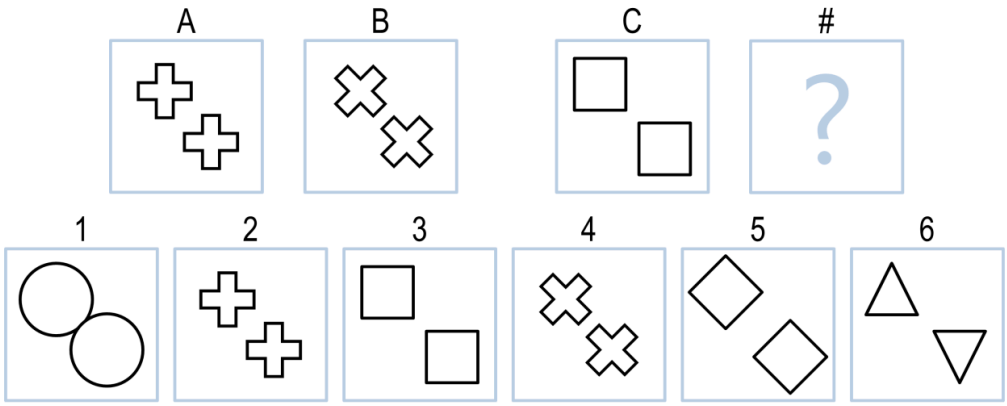
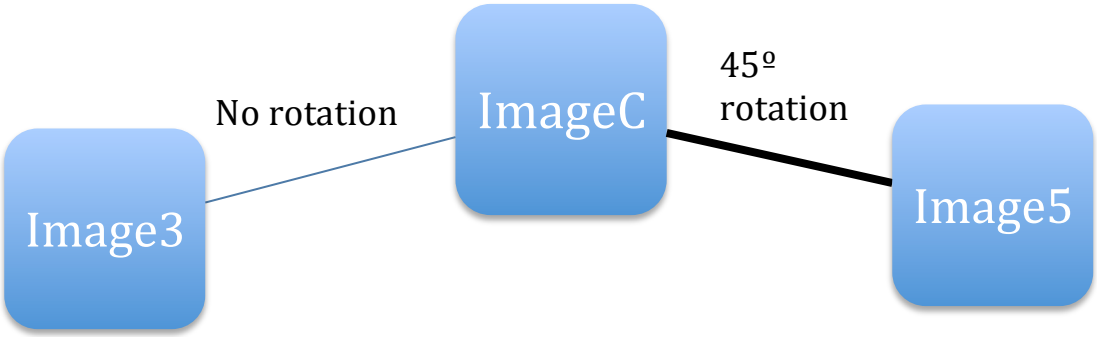


Image A = { Object1(Plus, Upper-Left, 30x30), Object2(Plus, Lower-Right, 30x30) }

Image B = { Object1(Plus, Upper-Left, 30x30), Object2(Plus, Lower-Right, 30x30) }

These are just example data structures that could be used to store each individual object in the image. However, they have stored the exact same information even though very clearly the two images A and B are very different. This is where the Analogy function comes in, which takes the images pixel by pixel, and evaluates the difference between the two. For this situation, it would notice that the characteristics are the same, however the pixels are slightly different – for in fact the only change between image A and B is that the two objects were rotated 45 degrees to the right. This Analogy function is run at the end as a last-case scenario because many images can be eliminated without it. For example, because A and B both have the same two objects in each image, we know for C that images 1,2,4, and 6 could never be it's match – they are immediately eliminated! This saves time complexity without having too run a very costly function as evaluating each pixel in an image. So now for C, we can now create its semantic network:



This network is extremely simple and also will lead to the answer through weighing the relationships between the starting point and possible answers. Image5 is weighted more than Image3 because of the relationship matching similarities between ImageA and ImageB. If this network was much bigger, let's say there were multiple steps in the process of finding the goal, then the path with the higher weight at each point would be chosen first, and if a goal was not found, backtracking would occur recursively until a new path found a goal. This weight is found by running the Analogy function between the object descriptions of the comparison, and then is evenly distributed between all of the possibilities. The percentage of matching mapped back to the example pair is then distributed among all of the possible choices. The key to this semantic network is the pre-processing step, where many choices can be eliminated if we know the answer has to contain certain types or sizes of objects.

Because of pre-processing, we are able to eliminate many possible choices that allow us to build a simpler semantic network that has higher probabilities of containing the right answer. After that, an Analogy function is used to really understand the relationships between two images, which is how the network is built. From there, we can simulate through the example pair and try to emulate that through weighting the links of higher priority to try, and then images of lower priority.