Adam Kmet B00095430
Fuzzy Logic

Temperature (inputVariable1)

For the measurement of the temperature, I have used 5 different levels of measurements. The temperature was measured in the range from 0 to 40. The levels are Too cold, cold, warm, hot and too hot.

Too cold = 0 ,0 , 10
Cold = 5,10 ,18
Warm=15 ,20 ,25
Hot=20 ,30, 35
Too hot=30, 40, 40

The reason why I went with this set up is because I believe the distribution of the temperature is very equal across the levels and is very well covering all the different settings of temperature. The levels are virtually divided into a low (too cold and cold) and a high (hot and too hot) range of temperatures with warm being ideal room temperature.

They are then mirrored with the breaking point being 18-20 degrees because it is the halfway point. I have used 18 degrees for the top range of cold because it was reacting the best to initial turning on of the AC as the default temperature on the AC is 18.

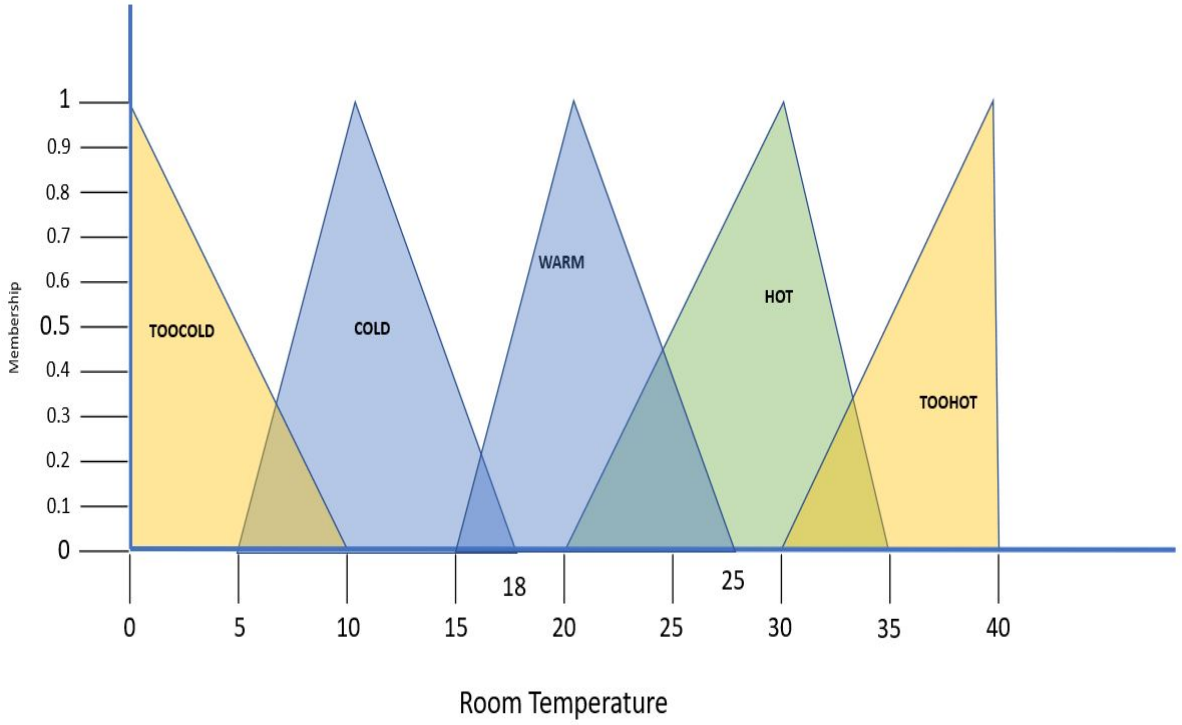
To make this a fuzzy problem/solution I made sure that the temperatures are overlapping with some degrees meaning that for example when it is 7 degrees, it can either mean that the temperature is too cold or just cold, or 32 degrees in room temperature can be labelled as hot or too hot.

The reason why the first and last label "too cold and too hot" have the first or last two values the same 0,0 and 40, 40, is because of the logical value of the bale itself. By this, I mean that when the temperature is too cold, most of the temperature will be rotating in a very small range and most likely closer to 0 or other way around for too hot will be very close to 40. The next level the starts at 5 which means that common degrees for too cold and cold will be from 5 to 10, therefore, I have decided to really make it obvious as to what too cold really is I kept 2 values of too cold in its triangle as 0,0. It would have same effect if I kept the second 0 as 1,2,3 or 4. The same would be applied for too hot

Temperature Shapes Triangle

For the temperature, I have used the Triangle shapes. The reason why is because the AC was reacting the best and fastest to any changes at any temperature. With the trapezoid shape, the AC was working also good but I was having major problems with getting out of some values. The AC would get stuck on some temperature for a few seconds then go wrong way ie. up instead of down but then eventually reaching the target temperature. This made me then try triangles and after implementation, I could see better results straight away.

Diagram for temperatures



Target inputvariable2

The range of the targets was also set from 0 to 40. The labels that I have picked to be were
Vlow= 0, 5, 10
low=5, 10, 12.5, 17.5
normal=12.5, 17.5, 22.5, 25
High=22.5, 25, 30, 35
Vhigh=30, 35, 37.5, 40

For the targets, I have tried to implement the temperature values to be as close as possible to the possible room temperatures because they should be the same in most cases. With the target value, the user is basically requesting the temperature and for easy implementation, the temperature requested should be in the same area as in the room temperature range.

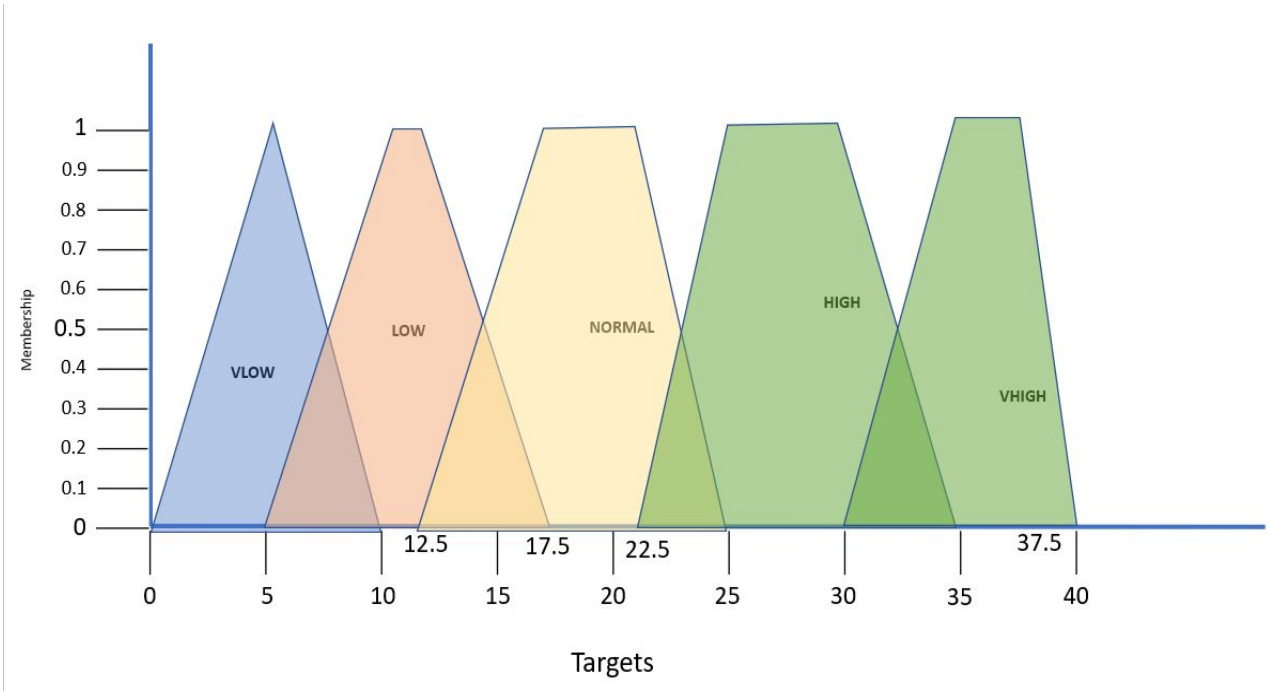
Because the values of the target are not as important as the temperature which is more of a guideline that is being followed, I was allowed to use different shapes, add more values to the shapes and also didn't have to start first and last value with the same 2 numbers (0,0 and 40,40). After small testing, I realised that this was not affecting the result and efficiency of the AC and therefore I changed it a bit.

Shapes Trapezoid

For the target temperature shape, I have picked Trapezoid. The reason why is because I believe that when the user is picking some value there should be more of range to the shape

than triangle has to offer. A triangle pinpoints the value almost exactly but trapezoid allows the value to be in some range(bigger range at least).

Diagram Target



Command range

The range of the command was to be set from -10 to 10 degrees and so I have picked again 5 different labels for the temperatures.
Vlow=-10 ,-8.5, -6.5 ,-4.5
Low=-6.5 ,-4.5 ,-2 ,0
Normal=-1, 0, 1
High=0,2 , 4.5 ,6.5
Vhigh=4.5, 6.5, 8.5, 10

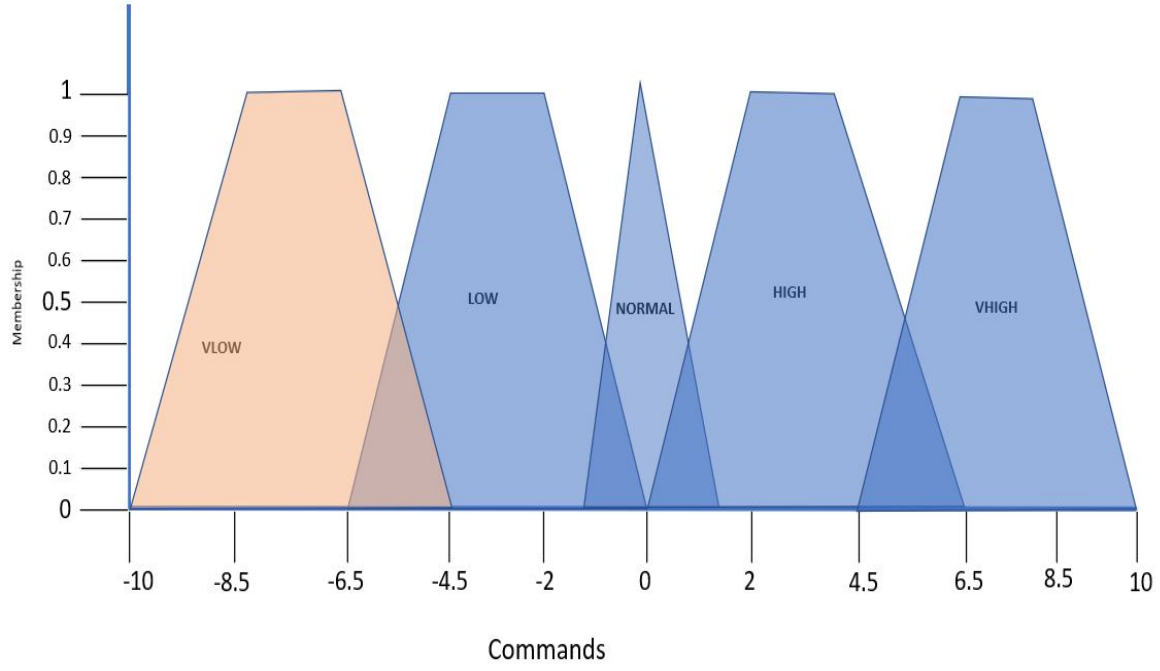
The reason why I have implemented 5 labels for the command temperature is because for a long time of testing I have been having trouble implementing the design that would work with 3, mainly because of the big jumps it would have to make. The program was working and was giving roughly correct temperature but was always slightly off and not too reliable.
The 5 label implementation allows me to make small changes at every step and slowly progress through the ranks of temperature as it goes. The ranks are exactly divided in half with breaking point being the 0. Normal label has the smallest range to it because it is acting

like a no changing variable. For example, if your room temperature is 25 degrees and you want it to be 25 then you set the label to normal and let it run 1 degree on and off the target temperature.

Shapes- Mix

The best working shapes for the commands I have figured out were Trapezoids. Same reason as before, because it offers a bigger range of movement when locating the user's target temperature. The middle part of the range (-1 to 1) I have decided to pick a triangle because it offers the ability to pinpoint the exact value and because the range of this triangle will be small its the perfect solution to make no or very little changes to the value.

Diagram for Commands



Rules and Matrix

target	too cold	cold	warm	hot	too hot
very low	normal	vlow	vlow	vlow	vlow
low	high	norm	low	low	vlow
normal	vhigh	vhigh	normal	vlow	low
high	vhigh	high	high	normal	low
very high	vhigh	vhigh	high	high	normal

The above matrix is displaying how I have implemented the rules for the AC system. In yellow I am displaying the room temperature and in red the target temperature. In blue the ac command is inserted to manipulate the room temperature.
The following rules have been made as a result of the matrix implementation

```
 //low end ( 0 to 18)
        ruleBlock.addRule(Rule.parse("if (temperature is warm) and
(target is low) then command is low", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is cold) and
(target is normal) then command is vhigh", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is cold) and
(target is low) then command is normal", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is cold) and
(target is vlow) then command is vlow", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is toocold) and
(target is vlow) then command is normal", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is toocold) and
(target is low) then command is high", engine));
        //high end (19 to 40)#
        ruleBlock.addRule(Rule.parse("if (temperature is warm) and
(target is high) then command is high", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is hot) and
(target is normal) then command is vlow", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is hot) and
(target is vhigh) then command is vhigh", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is toohot) and
(target is high) then command is low", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is toohot) and
(target is normal) then command is low", engine));

        ruleBlock.addRule(Rule.parse("if (temperature is cold) and
(target is vhigh) then command is vhigh", engine));
        ruleBlock.addRule(Rule.parse("if (temperature is toohot) and
(target is low) then command is vlow", engine));
```

Because I have tried to implement the commands with a small difference in the temperature I am able then to decrease the amount of the rules to 13 from 20, needed to make the AC work with good efficiency.
I am able to only reference in most of the rules the next up or down the temperature label for example if the current temperature is Toocold and the user wants it to be Toohot, I don't require to make a rule saying this but instead, I only have to reference the next label so based on my rules I say if Toocold and target low go high then rule gets to cold from there it

reads another command if cold and command is normal (not vhigh but only one close to wanted temperature) and hence picks this to follow and so on all the way to the matching temperature.
Originally I had rule for every step but logically I have ended up with over 20 rules and the system wasn't fast enough and was getting confused as to what to do next.

Running Program