Dynamic Game Difficulty Controller: Fuzzy Inference System

IMAT3406: Fuzzy Logic and Knowledge Based Systems (AI)

1 Introduction

A "fuzzy inference system is a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules and fuzzy reasoning" (Jang, J.S.R., 1997). Jang (1997) also adds that its application has been successful "in a wide variety of fields, such as automatic control, data classification" and "pattern recognition". With the fuzzy inference system(FIS) that I am proposing to develop; a dynamic game difficulty controller; the application of a fuzzy inference system could be beneficial in taking numerous "fuzzy" inputs that naturally exist within a game and thereby producing an output to control a games difficulty dynamically.

2 Literature Review

2.1 Mamdani vs Sugeno Systems

It's worth noting that the software used to develop this system will be 'MATLAB', using the 'Fuzzy Logic Toolbox' extension. This software only provides the opportunity to "implement two types of [FIS] is the toolbox: Mamdani and Sugeno" (Uk.mathworks.com, 2017), therefore, I will only be looking at these two types of FIS, however, other types such as the Tsukamoto fuzzy model do exist.

2.1.1 Mamdani

"The most commonly used fuzzy inference technique is the so-called Mamdani method" (Iancu, I., 2012 citing Mamdani, E.H. and Assilian, S., 1975). Proposed by Ebrahim Mamdani in 1975, it was the "first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators" (Jang, J.S.R., 1997). As described by the MathWorks website (2017), who also develop the MATLAB program, this type of FIS "expects the output membership functions to be fuzzy sets".

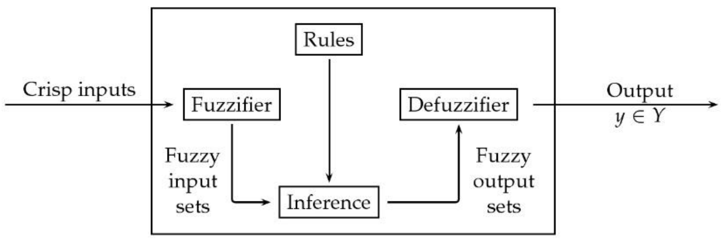
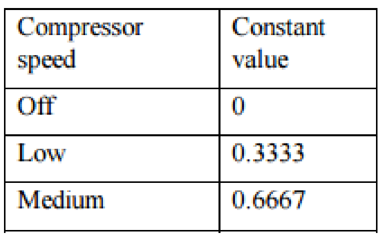


Figure 1

Fig. [1] (Boumella, Carlos and Iqbal, 2012) gives a graphical representation of the typical Mamdani FIS design, showcasing 4 steps: 1. Fuzzification of the input variables, 2. Rule evaluation, 3. Aggregation of the rule outputs and 4. Defuzzification.

2.1.2 Sugeno

The 'Sugeno' method of fuzzy inference whilst sharing many similar and perhaps identical processes with the Mamdani method (namely, the earlier stages of the process), there are



Compressor speed	Constant value
Off	0
Low	0.3333
Medium	0.6667

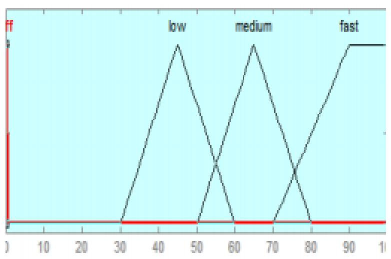


Figure 2 - Compressor speed membership functions

some differences. A "Sugeno output membership functions are either linear or constant" (Www-rohan.sdsu.edu, 2017). Kaur and Kaur, (2002), discuss the development process of a Sugeno-type FIS, where the output "can only either be constants or linear". They give an example of this (see Fig. [2] and [3]) where a compressor speed of an 'air-con' system is inferred using inputs from temperature and humidity sensors.

Figure 3 - Compressor speed membership functions

2.1.3 Comparison

Both systems are very much the same during the first few phases of the systems inference process, however, the forking point between the two exists at the final stage, where the "crisp output is generated from the fuzzy inputs" (Hamam and Georganas, 2008).

MathWorks (2017) lists the advantages for both the Mamdani and the Sugeno Method: -

Advantages of the Sugeno Method:

- Computationally efficient
- Works well with linear techniques (e.g. PID control)
- Works well with optimization and adaptive techniques
- It has a guaranteed continuity of the output surface
- Well suited to mathematical analysis

Advantages of the Mamdani Method:

- Intuitive
- Widespread acceptance
- Well suited to human input

From this we can gather that both system types are more suitable for a different type of application, perhaps where a more mathematical approach is required, use of the Sugeno system would be preferred. Hamam and Georganas studied the differences between the two systems in their paper which looked at the fuzzy evaluation of "quality of experience of Hapto-Audio-Visual applications". Both systems were built in MATLAB, using the same input and output variables. They note that the Sugeno system does not "defuzzify" the fuzzy output, but "uses a weighted average to compute the crisp output". In their results, they concluded that the Sugeno model "demonstrated higher accuracy and more dynamic values", whereas the Mamdani FIS displayed "consistency in the results".

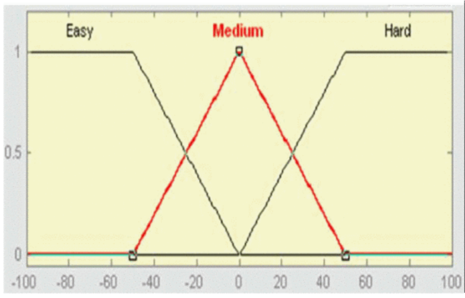
MathWorks suggestion that the Sugeno method "works well with optimization techniques" is also supported by Khosravanian (et al., 2017), where in his study of both FIS systems he found that a Sugeno-type FIS "can be more easily integrated with" …" a range of optimization algorithms", thereby "making it flexible enough the be adapted"; several other writers (Kaur and Kaur, 2012; Hamam and Georganas, 2008) also support this view. However, He also adds that a downside of the

Sugeno system its dependency on “quantity of data available” so the rule base and membership functions can be defined.

2.2 DYNAMIC/ADAPTIVE GAME DIFFICULTY

As suggested by Perez et al., (2015), “Players may cease from playing a chosen game sooner than expected for many reasons”. One of these reasons can often be the difficulty of the game, from both the perspective of it being “too easy” or “too hard”, thereby the player “becoming frustrated or bored”. An approach to fix this problem has been implemented to various degrees of success in games where the difficulty of a game is dynamically updated throughout to keep the player challenged, and therefore not become dissatisfied with the game.

A variety of systems exist to implement this idea, including those which use fuzzy concepts; three of these which use fuzzy concepts are discussed in works by “Perez et al., (2015)”, “Wang and Tseng, (2013)” and “Massoudi and Fassihi, (2013)”.



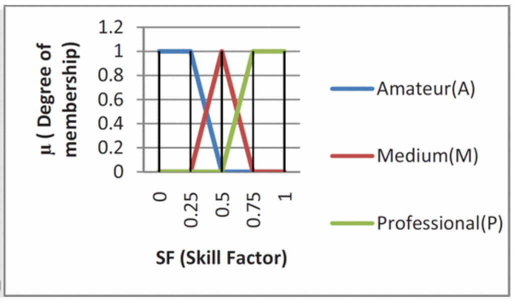


Figure 4- Difference of user HP

Figure 5-User skill factor

The latter two works use fuzzy concepts in a clearer way, but with both slightly different interpretations on what inputs/output is produced. Fig. [4] from Wang and Tseng (2013) shows how an input of the difference of two players health (HP) can be used to define three membership functions of game difficulty: Easy, Medium and Hard. On the other hand, Massoudi and Fassihi (Fig. [5]), (2013) “define a parameter named SF (Skill Factor)” (This is produced from an equation), which translates to three membership functions: Amateur, Medium and Professional.

Further benefits of using fuzzy concepts in this area are suggested as being “independence” and “scalability” by Perez et al., (2015). They go onto explain that difficulty can be adjusted “independently of explicit player feedback” and that as the number of “context variables and casual relationships increase [rules]”, more factors can be considered in the system which could lead to a better game experience. However, it could be argued that the scalability of the system does have a limit as a more complex system would take a greater degree of computing power, and therefore a longer period of computation, that may be incompatible with real-time difficulty adjustments in some games.

3 SYSTEM DESIGN

3.1 TYPE OF FIS

Looking back through the literature review at the two types of FIS, I ultimately decided upon developing my system using the Sugeno model. Whilst the argument for using either model can be made, I felt that the advantages of using the Sugeno system resonated to a greater degree with the type of application and context of my system.

In a real-life application, the FIS being developed would almost certainly be a smaller part or a whole (the actual game), because of this, I would infer that the system must be well optimised and efficient. A strong theme in the discussed literature was the Sugeno's systems advantage of being "Computationally efficient" (MathWorks, 2017) and "well optimised" (Khosravanian et al., 2017).

Furthermore, my decision to favour a Sugeno FIS over a Mamdani FIS also is supported by how the system will work in context to its application. This is due to the difference in output between the two systems, where the Sugeno-type FIS will use a "weighted average" to produce a crisp output. The benefit of this as studied by Hamam and Georganas (2008) is that the results when compared to a Mamdani system the results "demonstrated higher accuracy and more dynamic values". Given the mathematical nature of the system output, where the FIS controller will ultimately output a crisp mathematical value to give to the greater system (the entire game), a model that provided accurate and dynamic values would allow for the system to give a more certain and influential inference.

3.2 INPUTS

The three input variables I decided to use were:

- User Challenge Rating
- Session Playtime
- Deaths per Hour

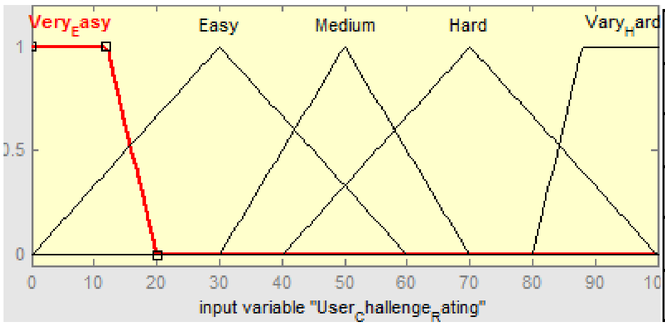
3.2.1 User Challenge Rating

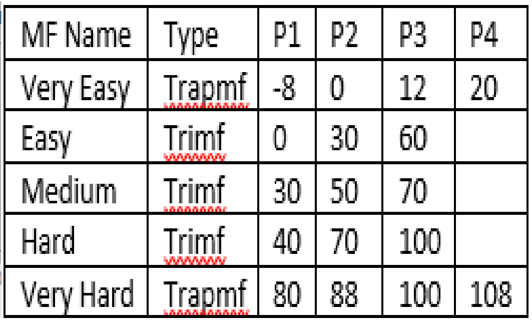
This input variable's main function is to help set and initial default idea on the output (difficulty). With appreciation for a real-world function, this input takes a value from 0 to 100, which would be expressed as a percentage. The input would be set in that it is chosen at the start of the game, with 100(%) being classed as the highest possible challenge rating.

The input has 5 membership functions(MF); Very Easy, Easy, Medium, Hard and finally Very Hard. My reasoning for implementing these 5 are due to both the idea that most games already gives the player an option at the start of a game which mimics these 5 values (however, sometimes the option to choose a set difficulty is not present in games or the number of options are reduced to 3 or 4), and the idea that players may be able to enjoy a game that they find easy/hard, and do not wish to adjust the difficulty too much (hence the more extreme area of very easy/hard).

The two extreme MF's (very easy/hard) are both trapezoidal with the plateau part of the shape stretching to the minimum and maximum range (as can be seen in Fig. [6]). This is because I felt that the input range where these two MF's preside are viewed as extreme values where the player would look for an overly easy or hard experience.

The other three MF's are triangular. Whilst (much like their extreme counterparts) the easy/hard MF's are symmetrical in position and area, the medium MF is slightly thinner in range covered. This is because when a player would select a difficulty, a "medium" difficulty would naturally represent a crisper value (still fuzzy however) due to its location being central. Whereas for example, it is somewhat likely that some players would consider a value of 51(%) to be "hard". The broader range given to the Easy and Hard MF's is due to many player's categorisation of game difficulty into the three central categories (as it is done with many games), and to represent to natural human fuzzy logic where a player's skill level will guide their opinions on the difficulty categorisation.





MF Name	Type	P1	P2	P3	P4
Very Easy	Trapmf	-8	0	12	20
Easy	Trimf	0	30	60	
Medium	Trimf	30	50	70	
Hard	Trimf	40	70	100	
Very Hard	Trapmf	80	88	100	108

Figure 6

Figure 7 – (P = Parameter)

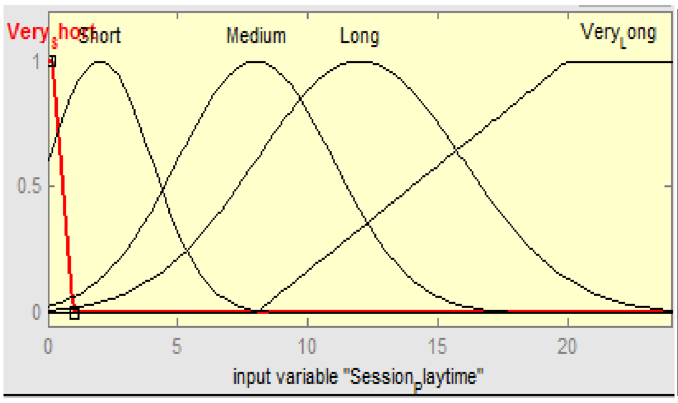
3.2.2 Session Playtime

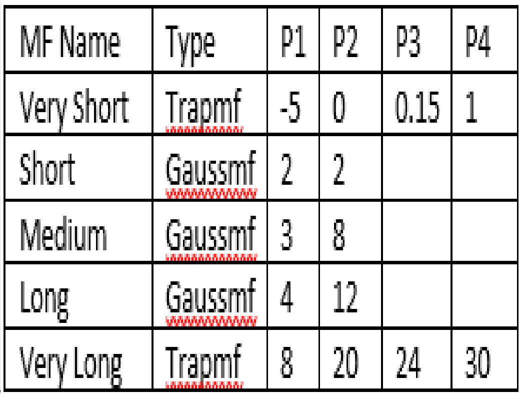
Simply, this input is equal to the player's total time played in hours (range from 0 to 24). This input maps the playtime to five MF's: Very Short, Short, Medium, Long and Very Long. Much like the input above, the first and last MF represent the most extreme categorisations of the input, with the inner three representing broader categories. The main function of this input is to help give weight to the rule set. Early on in a session the chosen difficulty rating (Section 3.2.1) should have a more weight in the output calculation, whilst as the session gets longer, the weight should reduce so the more dynamic input of "Deaths per Hour" (DpH) is given a greater weight in the rule set. This should functionally give the game/output a greater level of control and consistency in its computation, so the results early in a gaming session are not wildly different.

The Very Short MF is trapezoidal, however, whilst like the "Very Easy" mf above in that the lowest value (0) has a degree membership of 1, the plateau is much shorter (0.15) and it ends at 1. This is to represent the start of the game session, where the weight of the rules with this input are heavy.

The inner three MF's are all Gaussian MF's. They take two parameters, a 'σ' (standard deviation) and 'c' which represents the centre. The use of these MF types is due to a more fluid categorisation, where player's individual definitions of a short, medium and long playtime have a wider variance, whilst still having the majority of opinions on playtime centred in a similar range (e.g. 1σ away from centre).

The last MF is the Very Long categorisation, which again is trapezoidal. This MF has a much greater range than the others, to represent the extreme value. This is a design choice as most sessions won't last >10 hours, however, there is a small but group of people who will play a game for an extended period of time. The very long MF attempts to encapsulate this, with a wide plateau (Fig. [8]) at the end of the input range.





MF Name	Type	P1	P2	P3	P4
Very Short	Trapmf	-5	0	0.15	1
Short	Gaussmf	2	2		
Medium	Gaussmf	3	8		
Long	Gaussmf	4	12		
Very Long	Trapmf	8	20	24	30

Figure 8

Figure 9

3.2.3 Deaths per Hour

The final input variable is "Death's per Hour". This is just a numerical result of the equation:

$$\frac{Total\ Deaths}{Minutes\ Played}\ x\ 60$$

-with 60 representing 'an Hour' in minutes.

This is perhaps the only truly dynamic input variable that would be used in a real-world adaption of the system, where the input would normally be generated for the user based off other variables (Total Deaths and Session Playtime). However, for the sake of simplicity with this system, the variable is a starting input chosen by the user. This aspect of the input variable does stem from ideas discussed in the literature review (Section 2.2), where the benefit of variable "independence" allows values to be created without referring to explicit player feedback.

The four MF's in this input are as follows: None, Few, Some and Many. The MF of None is unique in this system as it is the only 'crisp' MF in the entire system. This MF is crisp because None represents a constant value of 0, and as such if the value is 0 the it will always belong in this MF (i.e. a membership grade of 1).

The other three MF's are all Gaussian types, and cover a range of 30 DpH, (which equates to a death every 2 minutes). The Few MF ranges from 0 to ~10, the Some MF's centre being slightly higher, but with a similar range. In contrast, the Many MF has a much bigger σ, which means it ranges from 0 to 30, however, as you can see from Fig. [10], both the Some and Few MF have a higher degree of membership nearer the lower values.

The main reason for the difference between few/some and many is the rate of deaths per minute that can be calculated. In terms of gameplay, dying 25 times per hour is not too different from dying 30 times per hour (dying every 2 minutes against every 2.4 minutes). However, when dying 5 times per hour and 10 times per hour (dying every 12 minutes against every 6 minutes), it is clear to see that there is a greater difference at the lower end.

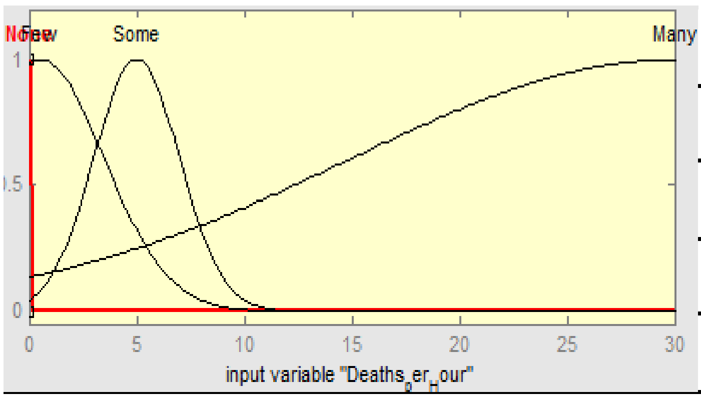
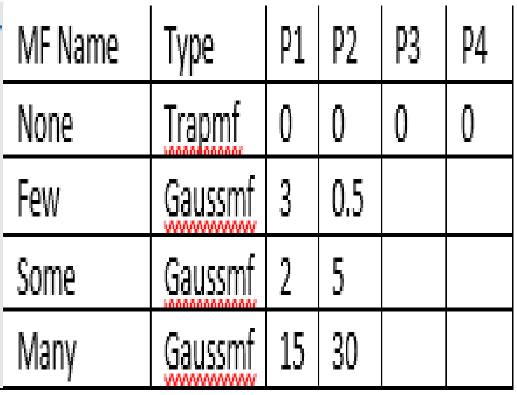


Figure 10



MF Name	Type	P1	P2	P3	P4
None	Trapmf	0	0	0	0
Few	Gaussmf	3	0.5		
Some	Gaussmf	2	5		
Many	Gaussmf	15	30		

Figure 11

3.3 Output

The output consists of three constant values that categorise the number of enemies that the player must face depending on the input values. The output values range from 5 to 20, with the MF's being: Few, Some and Many. One immediate effect of difficulty in games is the frequency of enemies, as it presents a different, yet harder challenge that in some respects is not achieved when the difficult of a game is raised by inflating enemy stats (such as damage or health), hence why this was the form of output variable that was chosen. This means that the easier overall difficulty's only present a few enemies (5-9), medium difficulties giving the player some enemies (10 – 14) and the harder difficulties presenting more enemies (15-20). This means that the system retains a positive aspect mentioned by Perez et al., (2015), where the system is "adaptable", and can also be scaled to fit into other systems.

3.4 Rule Base

There are a total of 27 rules in the rule base, with varying weights that affect the Output. A full list of the rules can be found in appendix A. Whilst each rule is different, they can be broadly defined into various groups:

3.4.1 Extreme Default Rules (1 and 2)

The first two rules (1 and 2) both have the maximum weight of 1 and look to set default values for the extreme difficulties, Very Easy and Very Hard. If the difficulty is Very Easy THEN the number of enemies is Few, but WHEN the difficulty is Very Hard THEN the number of enemies is Many. The reason for the maximum weight is to give the most extreme difficulties a more default high/low output value.

3.4.2 Default Time Rules

The following rules give the system some default output values based on the difficulty over time, ranging from the start (Very Short) to a Long play session. If the user difficulty is Easy, THEN there are few enemies. If the user difficulty is Medium, THEN there are Some enemies. Finally, if the difficulty is Hard, THEN there are many enemies.

3.4.2.1 Start (3:5)

All three of these have a weight of 0.8, to represent how at the start of a play session, the difficulty (No. of Enemies) must stay relatively consistent to the user defined difficulty, so no drastic output values are created.

3.4.2.2 Short, Medium and Long (6:8, 18:20, 21:23)

Fundamentally, these rules function in the same way as the Start Default rules, however, as the Play Session time input value increase from Short to Long, the weight of the rule decreases, so other rules which should dynamically change the difficulty have a greater effect. The weights for Short, Medium and Long Rules are 0.75, 0.5 and 0.3 respectively.

3.4.3 Default Difficulty/DpH Rules (9:11)

These three rules have a weight of 0.6, where they set the default amount of enemies for each user difficulty rating when the DpH value is also at "Some" (i.e. an Average amount of DpH). The purpose of this rule is to align the inference system to somewhat normalise the output is the DpH value is somewhat central, indicating that the player is experiencing a fair number of enemies for the difficulty they wanted. This links back to the idea of different players having a different preferred difficulty, and therefore this rule tries to keep the level of enemies consistent to make the enjoyment of the game high (by maintaining the DpH value).

3.4.4 Extreme Difficulty/Very Long Rules (24 and 25)

The purpose of these two rules, both with a weighting of 1, is to further reinforce the extreme difficulty rules set in rule 1 and 2. The rules here only look at when the play session is in the "Very Long" region, and then looks at the DpH value to set the Number of Enemies output.

3.4.5 Very Long Rules (16 and 17)

These two rules look to further define the outputs of the Easy and Hard difficulty when a long play session is underway. The Easy rule adds more enemies if the DpH is None (weighs 0.2), whereas the Hard rule sets the Enemies to Many, so the "Harder" difficulty is maintained throughout the play session.

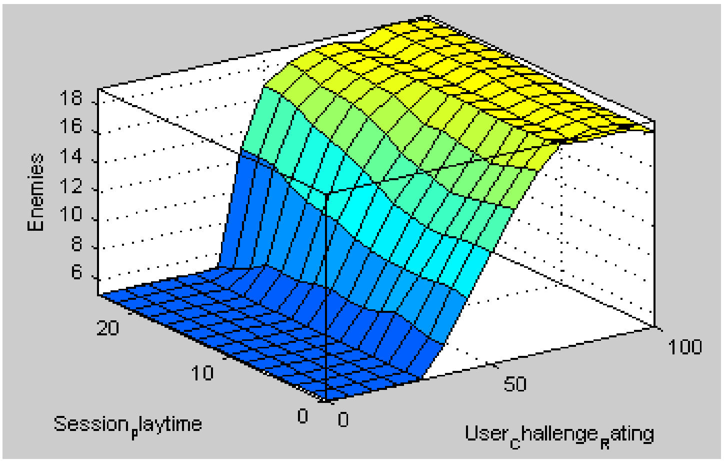
3.4.6 Medium/Medium Rules (26 and 27)

These two rules, both with a weight of 0.2, both attempt to slightly alter difficulty levels at the Medium difficulty during a "Medium" time played, so if the DpH is one of the two extremes (i.e. None or Many), the output is steered towards adjusting the difficulty in the appropriate direction.

3.4.7 Other Rules (12:15)

The final four rules again try to alter the number of enemies mostly dependant on the Easy/Hard difficulty and various DpH values. With a weight of 0.2, if the rating is Easy OR the DpH is many, the Enemies are Few. If the difficulty is Easy and the DpH is None, the enemies are raised to Some. If the difficulty is Hard, and the DpH is None, THEN the enemies are raised to Many with a 0.8 weight. Finally, if the difficulty is Hard, and the DpH is Few, THEN the enemies are again raised to Many, but with a slightly lower weighting of 0.6.

4 Testing and Evaluation



4.1 Surface Views (2 inputs)

These 3 Figures, (Fig. [12:14]) all show two inputs against each other. Fig. [12] shows the User Challenge Rating (UCR) input against the Session Playtime (SP). From this we can infer that generally, the UCR has the biggest impact on the Number of Enemies (NoE) output, however, there is a slight bump to the NoE when the UCR is >= "Medium" (~50) up to the Harder UCR values. It then levels out more in line with a lower SP output.

Figure 12

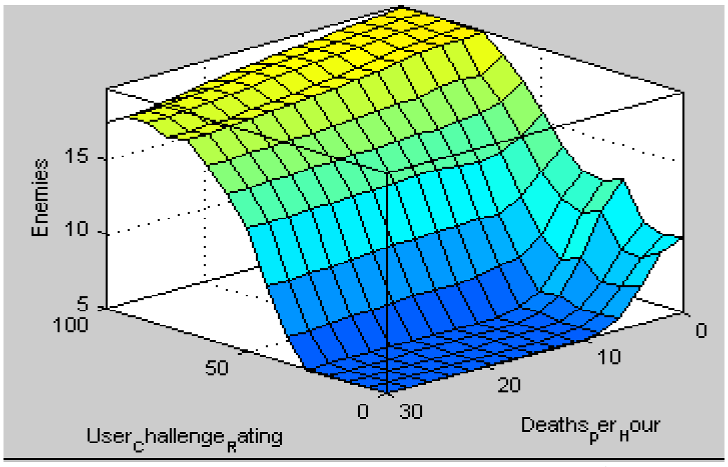
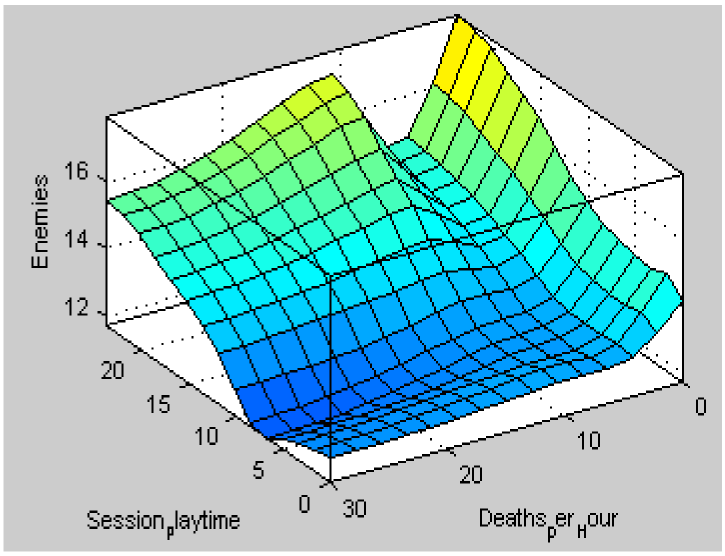


Fig. [13] puts the UCR against the DpH value. There is a much clearer effect on the output here compared to Fig. [12], where if the DpH is lower, the output is increase quite significantly compared to a higher DpH. This is expected, and it also showcases the rule where a "Very Low" UCR keeps the NoE lower to compensate for the requested "Extreme Difficulty".

Figure 13



The final figure for this section is the SP against the DpH values (Fig. [14]). In this surface view, a greater degree of variance is shown, indicating that these two inputs had more of a dynamic effect on the final output. What is particularly notable is the drop when the DpH goes between0 and 10, especially at a high SP. This could be due to the special rules that edit the output when the DpH is at "0", then only crisp value in the System.

Figure 14

4.2 Input Tests

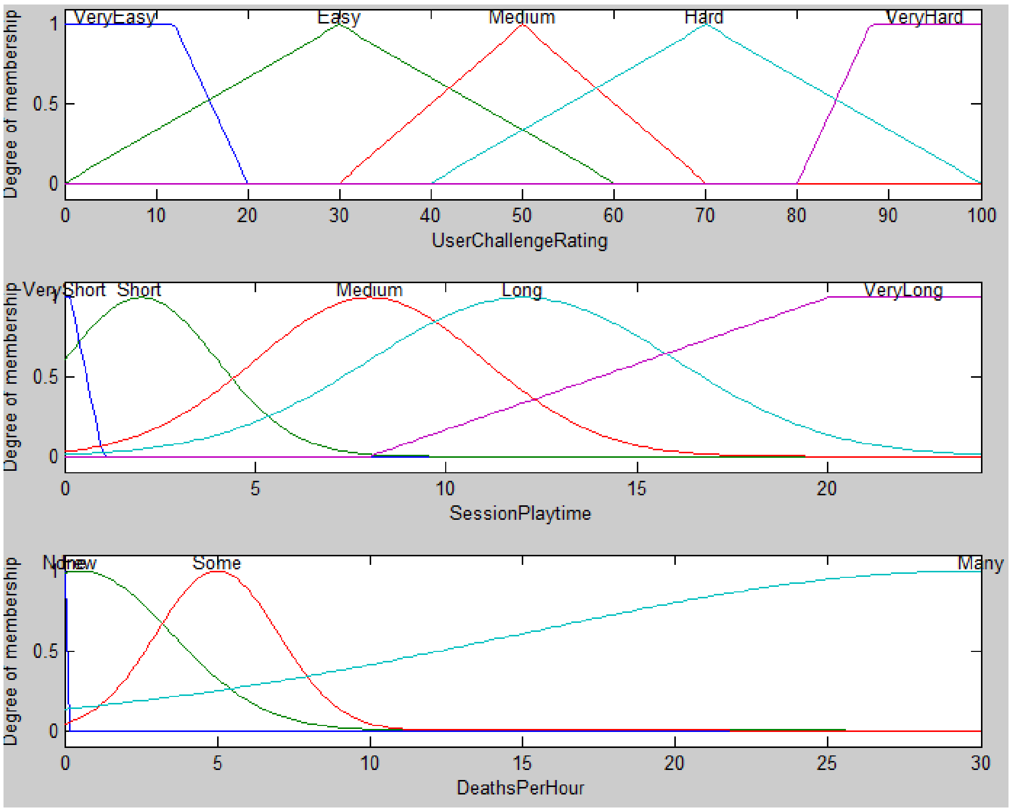


Figure 15

Fig. [15] Shows a plot of each input and its MF's. From top to bottom, there as follows: User Challenge Rating, Session Playtime and Deaths per Hour.

4.2.1 Test using Excel Input Data

Please See Appendix B for the full list of values used

Some notable results from this set of tests include:

```
22)  In(1): 86.00,  In(2)  21.00,  In(3)  5.00 => Out:  19.70
```

Test 22 went in with and Input set of: 86% Difficulty, 21 Hours Played and 5 Deaths per Hour. It had an output of: 19.70 enemies (ROUND TO 20).

We can see from the Rule Viewer (Appendix B1) that all the rules that fired and then hit the criteria given by the inputs added to the “Many” output Member Function, with no other rules firing that would’ve brought the Enemy Frequency output down. Overall, Rule 2 and Rule 24 seemed to have the most impact on the high output.

```
20)  In(1): 67.00,  In(2)  8.00,  In(3)  13.00 => Out:  17.82
```

Test 20 went in with and Input set of: 67% Difficulty, 8 Hours Played and 13 Deaths per Hour. It had an output of: 17.82 enemies (ROUND TO 18).

Despite a relatively low User Difficulty at 67% (Still inside the Medium Bracket), this set of input produced a very high output. We can see from the Rule Viewer (Appendix B2) that Rule 13 and Rule 20 had the greatest effect in terms of pushing the output to a high number. This is because Rule 13 fires with Input 1 being very close to having a membership degree of 1, and despite it being only weighed to 0.6, it is one of the few rules that fires whilst hitting the criteria (one of 4 in fact).

```
17)  In(1): 2.00,  In(2)  3.00,  In(3)  17.00 => Out:  5.00
```

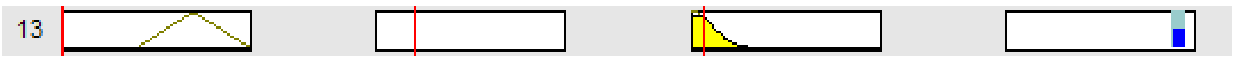
Test 17 went in with and Input set of: 2% Difficulty, 3 Hours Played and 17 Deaths per Hour. It had an output of: 5 enemies.

Test 17 gave us the lowest possible output available in the system process. Appendix B3 shows us that this was almost exclusively due to Rule 1, which states that “IF input1 IS “VERY EASY” THEN output1 IS “FEW”. Being that the input set did not fire any other rules that its value had much (if any) impact on, the weight of Rule 1 was enough to keep the output to a minimum.

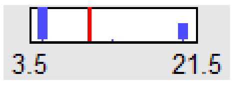
```
1)  In(1): 0.00,  In(2)  5.00,  In(3)  2.00 => Out:  10.08
```

Test 1 went in with and Input set of: 0% Difficulty, 5 Hours Played and 2 Deaths per Hour. It had an output of: 10.08 enemies (ROUND TO 10).

Test 1 could be considered either and outlier or a fault in the system. Appendix B4 shows that whilst much like Test 17, Rule 1 is very much the dominant fired rule. However, despite similar input values, the difference in output is very clear. If we examine the Rule Viewer in more detail, we can see the reason for the split is from Input 3 being so high.



Unlike Test 17, rule 13 has a huge impact on the final output. Rule 13 is as follows: IF input1 OR input3 THEN output1 IS “MANY”. We saw this rule fired in test 20 where it had a big effect on the output, however, with this input set, we can see the rules that are fired are very split between "Few" and "Many". This ultimately causes the consequents to split the output weight and the FIS interprets a value in the middle, (10.08).



5 Critical Analysis and Conclusion

Overall I think the performance of my system was successful. I originally made the choice to go with a Sugeno FIS, which would return a crisp output value, rather than the output of a Mamdani system that would return a fuzzy set group. I think that this choice was justified within the context of my system, as it is a fuzzy controller system, and in a real-life application this system would ultimately work as part of a whole. It could therefore be suggested that producing a crisp output would be more beneficial to a Gameplay System, as work in the Literature Review (Section 2.2) showed that the benefits of using fuzzy logic within a game allow for "scalability", "independence" and "adaptability" (Perez et al., 2015). We could consider the system to be partially independent when taking values that are generated automatically by the game (Deaths per Hour), we could consider the system somewhat adaptable to replace variables to fit a different game/genre, and we could consider the system somewhat scalable. We further found that using Sugeno over Mamdani meant some key advantages such as being "Computationally Efficient", "Works well with optimisations" and is "well suited to mathematical analysis".

On the other hand, the system was still quite simple, and could be considered more of a prototype of using FIS's for adaptive video game difficulty (AVGD). Firstly, the use of only 3 input could be considered to be weakness of the system, both in terms of a lack of complexity and a lack of true testing of a AVGD FIS. In a real scenario, the system would have to use more than three inputs as the actual complexity of current games demands it. If this was to be addressed however, the systems performance could come into question, as although a more efficient FIS-type was used, the true maximum capabilities of the system are unknown.

It could also be argued that whilst we "simulated" independent data, in reality the system would have to use multiple inputs to outputs to inputs, and the simulation of data that would be calculated by system would provide fairly accurate results, again the performance of the system is an unknown.

Despite the small amount of inputs used, the testing of the system did provide both interesting and distinctive results. Fig. [14] was able to clearly showcase an impact to a potential input set where the "Number of Enemies" was reduced where Deaths per Hour were low, but not 0. In addition, testing of individual input sets showed that some tests would provide interesting results, such as test 20, which showed that with the right conditions, a desired outcome of a dynamic, changing difficulty could be possible.

Whilst many of the tests provided outputs that were expected, a few input sets did produce what can be considered "false" results (within the system context). Test 1 in Appendix B showed this, as a two rules were fired with a considerable weight, so upon translating the fuzzy sets into a crisp output value, the FIS returned a value that was in the middle of the two. This, technically, was the correct process and result, however, looking at the greater context and when comparing the result with Test 17 in particular, it can be assumed that there is a fault in the system with the rule set, where an input (in this case input3, Deaths per Hour) has too much influence on the output.

Overall, the system provided a solid groundwork that demonstrates potential in applying fuzzy logic and a FIS to an Adaptive Video Game Difficulty system. There are clear benefits of using the Sugeno model for this type of application, with the AVGD system being benefitted from the use of fuzzy concepts. However, the system produced, whilst giving some very good data and clear dynamic values, has its faults, namely some faults with its current rule system and some wider criticisms with regards to potential performance on similar, but more complex FIS.

6 References

Jang, J.S.R., Sun, C.T. and Mizutani, E., 1997. Fuzzy inference systems. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*, pp.73-91.

Uk.mathworks.com. (2017). *Types of Fuzzy Inference Systems - MATLAB & Simulink*. [online] Available at: https://uk.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html?s_tid=gn_loc_drop.

Iancu, I., 2012. *A Mamdani type fuzzy logic controller*. Rijeka: INTECH Open Access Publisher.

Boumella, N., Carlos, J. and Iqbal, S. (2012). Enhancing Fuzzy Controllers Using Generalized Orthogonality Principle. *Fuzzy Controllers- Recent Advances in Theory and Applications*.

Www-rohan.sdsu.edu. (2017). *Tutorial (Fuzzy Logic Toolbox)*. [online] Available at: http://www-rohan.sdsu.edu/doc/matlab/toolbox/fuzzy/fuzzyt27.html.

Kaur, A. and Kaur, A. (2002). *Comparison of Mamdani-Type and Sugeno-Type Fuzzy Inference Systems for Air Conditioning System*. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.486.1238&rep=rep1&type=pdf.

Hamam, A. and Georganas, N. (2008). A comparison of Mamdani and Sugeno fuzzy inference systems for evaluating the quality of experience of Hapto-Audio-Visual applications. *2008 IEEE International Workshop on Haptic Audio visual Environments and Games*.

Khosravanian, R., Sabah, M., Wood, D. and Shahryari, A. (2017). *Weight on drill bit prediction models: Sugeno-type and Mamdani-type fuzzy inference systems compared*.

Perez, L., Calla, L., Valente, L., Montenegro, A. and Clua, E. (2015). Dynamic Game Difficulty Balancing in Real Time Using Evolutionary Fuzzy Cognitive Maps. *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*.

Wang, J. and Tseng, Y. (2013). Dynamic difficulty adjustment by fuzzy rules using in a neural network controlled game. *2013 Ninth International Conference on Natural Computation (ICNC)*.

Massoudi, P. and Fassihi, A. (2013). Achieving dynamic AI difficulty by using reinforcement learning and fuzzy logic skill metering. *2013 IEEE International Games Innovation Conference (IGIC)*.

7 Bibliography

Mendel, J.M., Martin, M.A., & Sophomore, M.A., 2002. *Flirtation, a Very Fuzzy Prospect: a Flirtation Advisor.*

Mamdani, E.H. and Assilian, S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, *7*(1), pp.1-13.

Vrusias, B. (2006). *Fuzzy Logic 3*.

Yusoff, M., Mutalib, S., Rahman, S. and Mohamed, A. (2007). Intelligent Water Dispersal Controller: Comparison between Mamdani and Sugeno Approaches. *2007 International Conference on Computational Science and its Applications (ICCSA 2007)*.

Guney, K. and Sarikaya, N. (2017). *COMPARISON OF MAMDANI AND SUGENO FUZZY INFERENCE SYSTEM MODELS FOR RESONANT FREQUENCY CALCULATION OF RECTANGULAR MICROSTRIP ANTENNAS*.

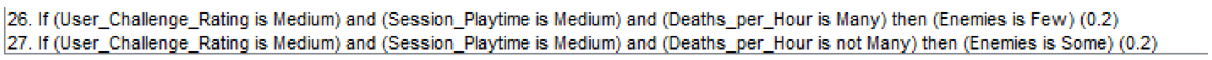
Vukic, Z. and Velagic, J. (1999). *COMPARATIVE ANALYSIS OF MAMDANI AND SUGENO TYPE FUZZY AUTOPILOTS FOR SHIPS*. 1st ed. IEEE / Institute of Electrical and Electronics Engineers Incorporated.

Rollings, A. and Adams, E. (2003). *Andrew Rollings and Ernest Adams on game design*. 1st ed. Indianapolis, Ind.: New Riders, pp.200-239.

Hunicke, R. (2005). The case for dynamic difficulty adjustment in games. *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology - ACE '05*.

8 APPENDIX

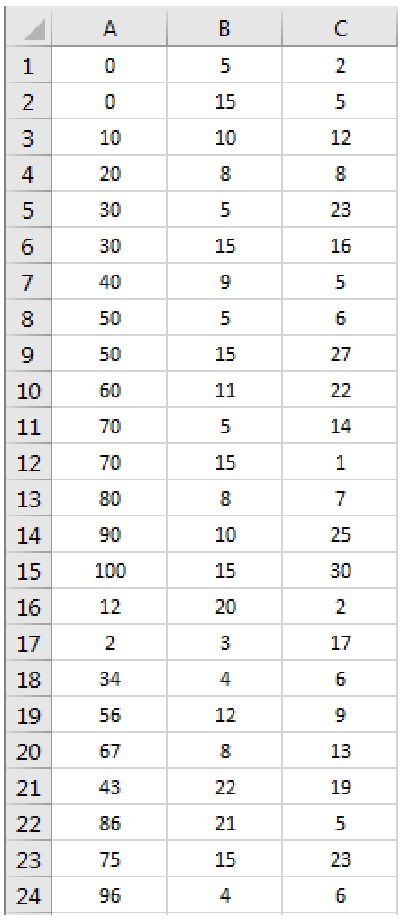
8.1 A



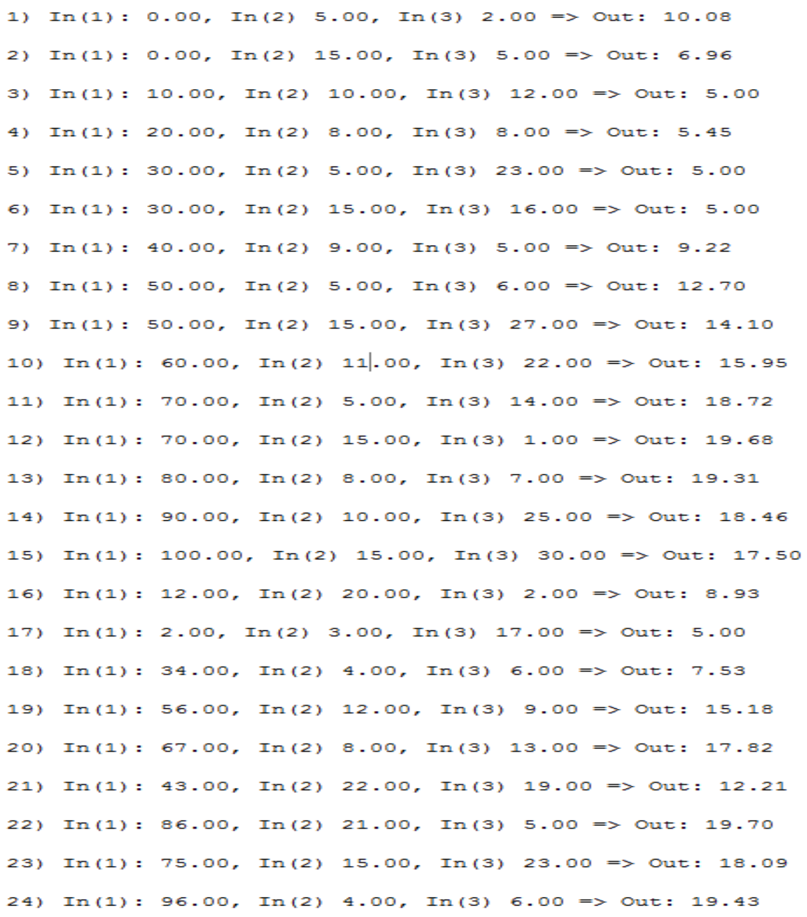
```
26. If (User_Challenge_Rating is Medium) and (Session_Playtime is Medium) and (Deaths_per_Hour is Many) then (Enemies is Few) (0.2)
27. If (User_Challenge_Rating is Medium) and (Session_Playtime is Medium) and (Deaths_per_Hour is not Many) then (Enemies is Some) (0.2)
```

1. If (User_Challenge_Rating is Very_Easy) then (Enemies is Few) (1)
2. If (User_Challenge_Rating is Very_Hard) then (Enemies is Many) (1)
3. If (User_Challenge_Rating is Easy) and (Session_Playtime is Very_Short) then (Enemies is Few) (0.8)
4. If (User_Challenge_Rating is Hard) and (Session_Playtime is Very_Short) then (Enemies is Many) (0.8)
5. If (User_Challenge_Rating is Medium) and (Session_Playtime is Very_Short) then (Enemies is Some) (0.8)
6. If (User_Challenge_Rating is Easy) and (Session_Playtime is Short) then (Enemies is Few) (0.75)
7. If (User_Challenge_Rating is Medium) and (Session_Playtime is Short) then (Enemies is Some) (0.75)
8. If (User_Challenge_Rating is Hard) and (Session_Playtime is Short) then (Enemies is Many) (0.75)
9. If (User_Challenge_Rating is Easy) and (Deaths_per_Hour is Some) then (Enemies is Few) (0.6)
10. If (User_Challenge_Rating is Medium) and (Deaths_per_Hour is Some) then (Enemies is Some) (0.6)
11. If (User_Challenge_Rating is Hard) and (Deaths_per_Hour is Some) then (Enemies is Many) (0.6)
12. If (User_Challenge_Rating is Easy) or (Deaths_per_Hour is Many) then (Enemies is Few) (0.2)
13. If (User_Challenge_Rating is Hard) or (Deaths_per_Hour is Few) then (Enemies is Many) (0.6)
14. If (User_Challenge_Rating is Hard) and (Deaths_per_Hour is None) then (Enemies is Many) (0.8)
15. If (User_Challenge_Rating is Easy) and (Deaths_per_Hour is None) then (Enemies is Some) (0.6)
16. If (User_Challenge_Rating is Hard) and (Session_Playtime is Very_Long) then (Enemies is Many) (0.8)
17. If (User_Challenge_Rating is Easy) and (Session_Playtime is Very_Long) and (Deaths_per_Hour is None) then (Enemies is Many) (0.2)
18. If (User_Challenge_Rating is Easy) and (Session_Playtime is Medium) then (Enemies is Few) (0.5)
19. If (User_Challenge_Rating is Medium) and (Session_Playtime is Medium) then (Enemies is Some) (0.5)
20. If (User_Challenge_Rating is Hard) and (Session_Playtime is Medium) then (Enemies is Many) (0.5)
21. If (User_Challenge_Rating is Easy) and (Session_Playtime is Long) then (Enemies is Few) (0.3)
22. If (User_Challenge_Rating is Medium) and (Session_Playtime is Long) then (Enemies is Some) (0.3)
23. If (User_Challenge_Rating is Hard) and (Session_Playtime is Long) then (Enemies is Many) (0.3)
24. If (User_Challenge_Rating is Very_Hard) and (Session_Playtime is Very_Long) and (Deaths_per_Hour is not Many) then (Enemies is Many) (1)
25. If (User_Challenge_Rating is Very_Easy) and (Session_Playtime is Very_Long) and (Deaths_per_Hour is Many) then (Enemies is Few) (1)

8.2 B

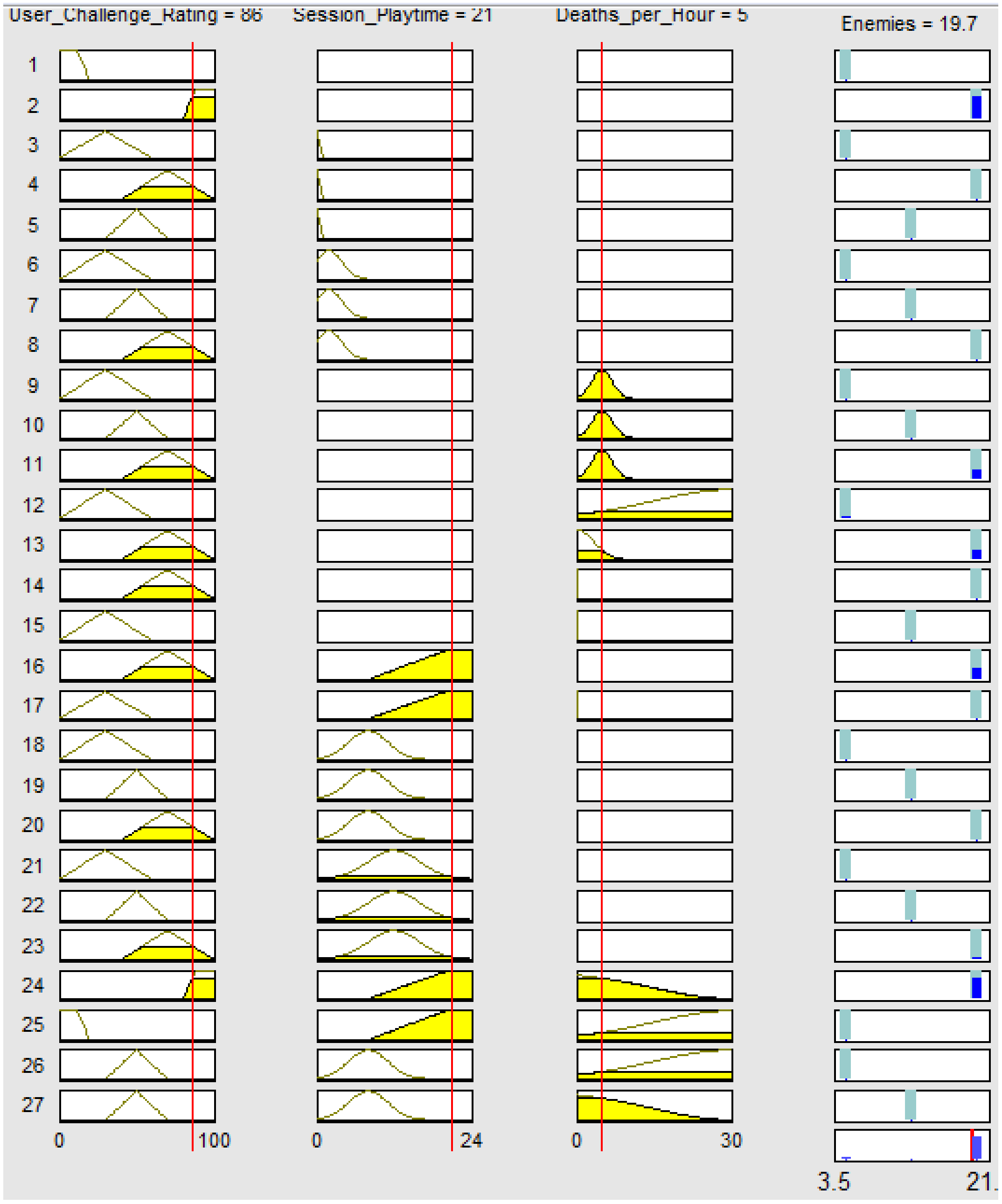


	A	B	C
1	0	5	2
2	0	15	5
3	10	10	12
4	20	8	8
5	30	5	23
6	30	15	16
7	40	9	5
8	50	5	6
9	50	15	27
10	60	11	22
11	70	5	14
12	70	15	1
13	80	8	7
14	90	10	25
15	100	15	30
16	12	20	2
17	2	3	17
18	34	4	6
19	56	12	9
20	67	8	13
21	43	22	19
22	86	21	5
23	75	15	23
24	96	4	6

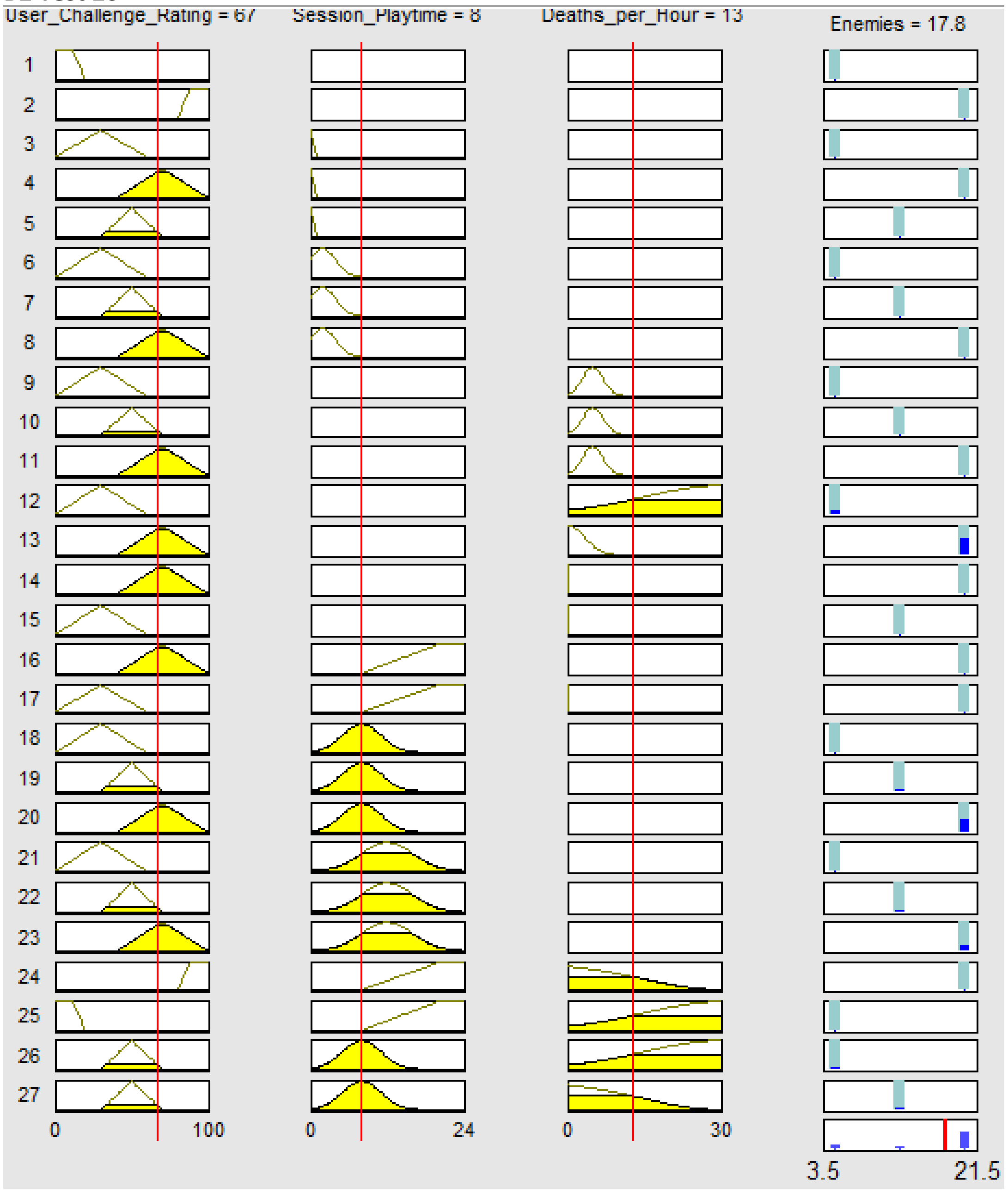


```
1) In(1): 0.00, In(2) 5.00, In(3) 2.00 => Out: 10.08
2) In(1): 0.00, In(2) 15.00, In(3) 5.00 => Out: 6.96
3) In(1): 10.00, In(2) 10.00, In(3) 12.00 => Out: 5.00
4) In(1): 20.00, In(2) 8.00, In(3) 8.00 => Out: 5.45
5) In(1): 30.00, In(2) 5.00, In(3) 23.00 => Out: 5.00
6) In(1): 30.00, In(2) 15.00, In(3) 16.00 => Out: 5.00
7) In(1): 40.00, In(2) 9.00, In(3) 5.00 => Out: 9.22
8) In(1): 50.00, In(2) 5.00, In(3) 6.00 => Out: 12.70
9) In(1): 50.00, In(2) 15.00, In(3) 27.00 => Out: 14.10
10) In(1): 60.00, In(2) 11.00, In(3) 22.00 => Out: 15.95
11) In(1): 70.00, In(2) 5.00, In(3) 14.00 => Out: 18.72
12) In(1): 70.00, In(2) 15.00, In(3) 1.00 => Out: 19.68
13) In(1): 80.00, In(2) 8.00, In(3) 7.00 => Out: 19.31
14) In(1): 90.00, In(2) 10.00, In(3) 25.00 => Out: 18.46
15) In(1): 100.00, In(2) 15.00, In(3) 30.00 => Out: 17.50
16) In(1): 12.00, In(2) 20.00, In(3) 2.00 => Out: 8.93
17) In(1): 2.00, In(2) 3.00, In(3) 17.00 => Out: 5.00
18) In(1): 34.00, In(2) 4.00, In(3) 6.00 => Out: 7.53
19) In(1): 56.00, In(2) 12.00, In(3) 9.00 => Out: 15.18
20) In(1): 67.00, In(2) 8.00, In(3) 13.00 => Out: 17.82
21) In(1): 43.00, In(2) 22.00, In(3) 19.00 => Out: 12.21
22) In(1): 86.00, In(2) 21.00, In(3) 5.00 => Out: 19.70
23) In(1): 75.00, In(2) 15.00, In(3) 23.00 => Out: 18.09
24) In(1): 96.00, In(2) 4.00, In(3) 6.00 => Out: 19.43
```

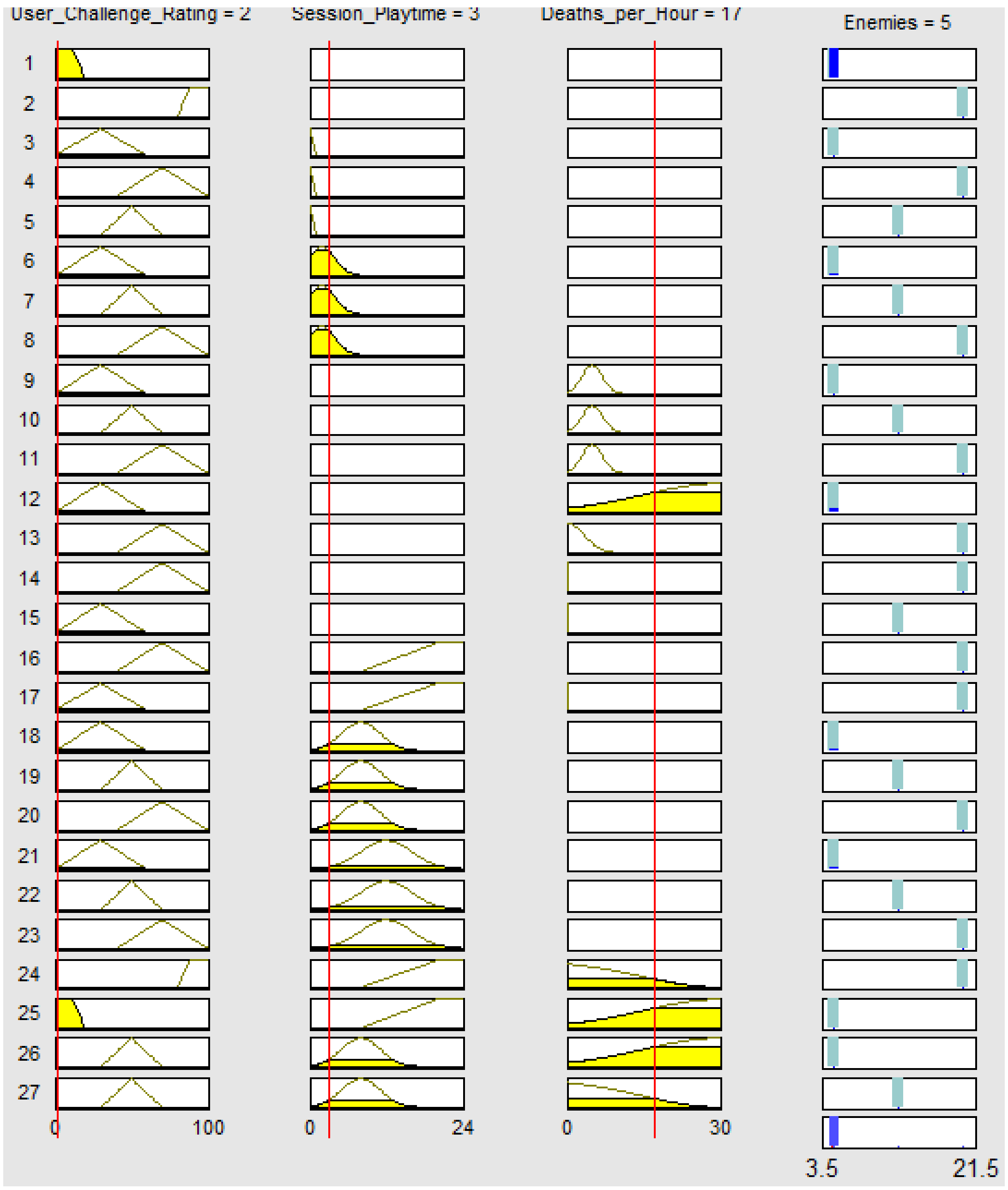
8.2.1 B1 Test 22



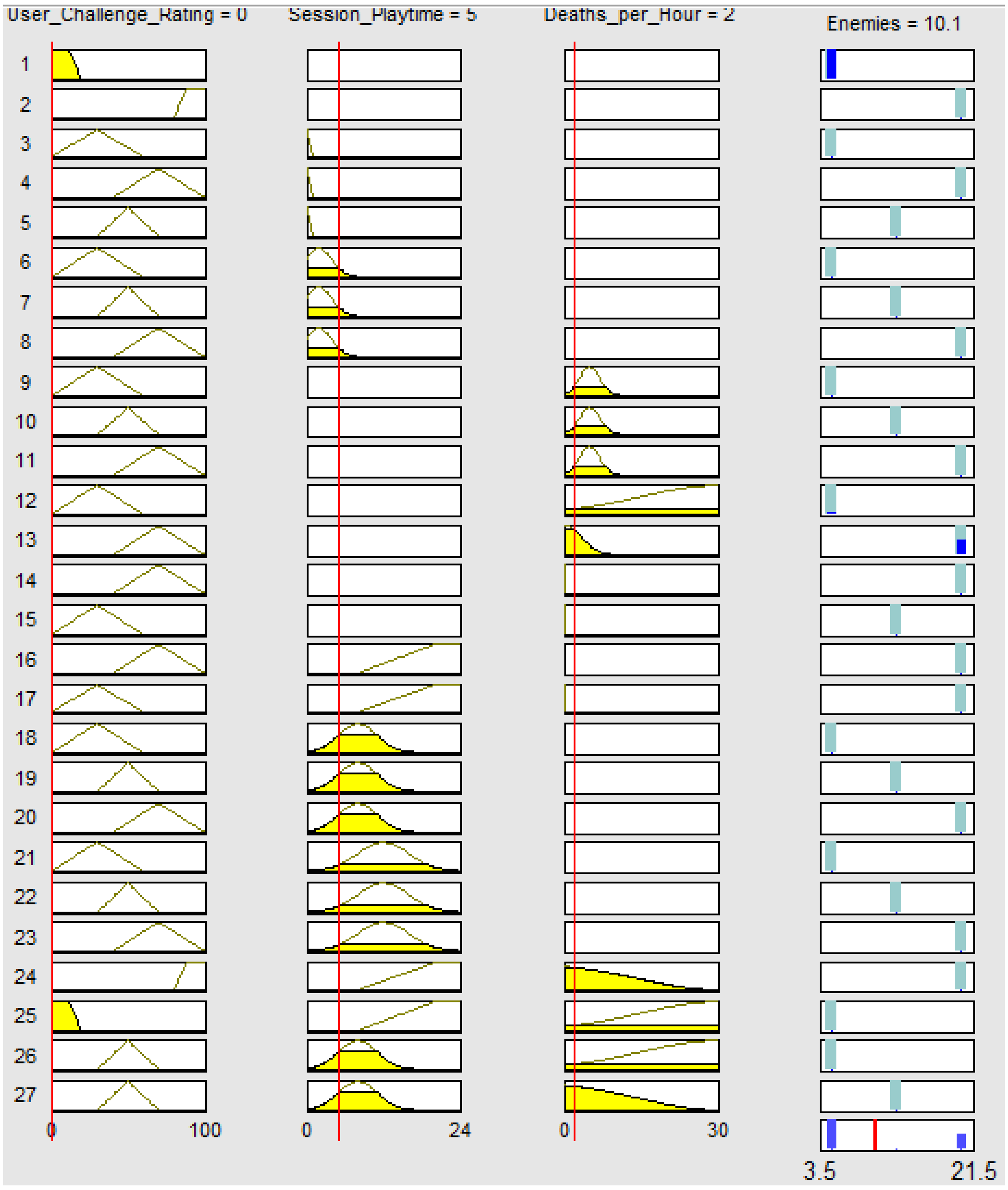
8.2.2 B2 Test 20



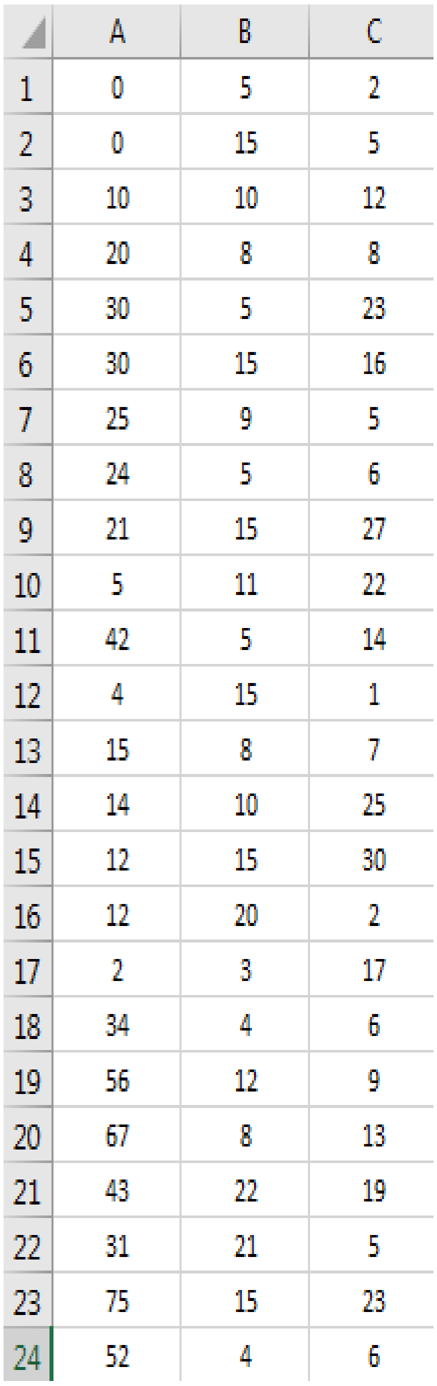
8.2.3 B3 Test 17


User_Challenge_Rating = 2
Session_Playtime = 3
Deaths_per_Hour = 17
Enemies = 5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
0
100
0
24
0
30
3.5
21.5

8.2.4 B4 Test 1


User_Challenge_Rating = 0
Session_Playtime = 5
Deaths_per_Hour = 2
Enemies = 10.1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
0
100
0
24
0
30
3.5
21.5

8.3 C



	A	B	C
1	0	5	2
2	0	15	5
3	10	10	12
4	20	8	8
5	30	5	23
6	30	15	16
7	25	9	5
8	24	5	6
9	21	15	27
10	5	11	22
11	42	5	14
12	4	15	1
13	15	8	7
14	14	10	25
15	12	15	30
16	12	20	2
17	2	3	17
18	34	4	6
19	56	12	9
20	67	8	13
21	43	22	19
22	31	21	5
23	75	15	23
24	52	4	6

```
1) In(1): 0.00, In(2) 5.00, In(3) 2.00 => Out: 10.08

2) In(1): 0.00, In(2) 15.00, In(3) 5.00 => Out: 6.96

3) In(1): 10.00, In(2) 10.00, In(3) 12.00 => Out: 5.00

4) In(1): 20.00, In(2) 8.00, In(3) 8.00 => Out: 5.45

5) In(1): 30.00, In(2) 5.00, In(3) 23.00 => Out: 5.00

6) In(1): 30.00, In(2) 15.00, In(3) 16.00 => Out: 5.00

7) In(1): 25.00, In(2) 9.00, In(3) 5.00 => Out: 6.94

8) In(1): 24.00, In(2) 5.00, In(3) 6.00 => Out: 6.23

9) In(1): 21.00, In(2) 15.00, In(3) 27.00 => Out: 5.00

10) In(1): 5.00, In(2) 11.00, In(3) 22.00 => Out: 5.00

11) In(1): 42.00, In(2) 5.00, In(3) 14.00 => Out: 9.38

12) In(1): 4.00, In(2) 15.00, In(3) 1.00 => Out: 9.60

13) In(1): 15.00, In(2) 8.00, In(3) 7.00 => Out: 5.58

14) In(1): 14.00, In(2) 10.00, In(3) 25.00 => Out: 5.00

15) In(1): 12.00, In(2) 15.00, In(3) 30.00 => Out: 5.00

16) In(1): 12.00, In(2) 20.00, In(3) 2.00 => Out: 8.93

17) In(1): 2.00, In(2) 3.00, In(3) 17.00 => Out: 5.00

18) In(1): 34.00, In(2) 4.00, In(3) 6.00 => Out: 7.53

19) In(1): 56.00, In(2) 12.00, In(3) 9.00 => Out: 15.18

20) In(1): 67.00, In(2) 8.00, In(3) 13.00 => Out: 17.82

21) In(1): 43.00, In(2) 22.00, In(3) 19.00 => Out: 12.21

22) In(1): 31.00, In(2) 21.00, In(3) 5.00 => Out: 8.14

23) In(1): 75.00, In(2) 15.00, In(3) 23.00 => Out: 18.09

24) In(1): 52.00, In(2) 4.00, In(3) 6.00 => Out: 13.45
```