**Fuzz in a Row**

**Application of fuzzy logic for a discrete decision tree**



**Stijn Verdenius**

**University of Amsterdam**

Abstract

Introduction

Methods

Result

Discussion

References

Appendix A

Appendix B

1. Introduction

Applications of fuzzy logic can be found within many fields of research. They are usually used for continuous variables, yet not an adequate amount of research has been done on the application of fuzzy logic in discrete problem settings (Pang, Wang, Zhou & Dong, 2004). In this paper fuzzy logic will be applied to a discrete decision problem: the game of “Connect-4”.

Games lend themselves well as testbed for AI applications. They form a simulation in where the inputs and outputs are easily distinguishable and the expected behavior is easily evaluated. This is not an unfamiliar idea, “Computational Intelligence methods have been employed to computer games since they provide dynamic and challenging elements that are similar to real world problems” according to Sahin and Kumbasar (2017). Yet it was never tried on a discrete game such as this. Thereupon an AI agent was developed in this paper using type-I fuzzy logic to play the game of Connect-4.

1.1 The game

Connect-4 is a game played by 2 players, let’s name them “X” and “O”. The game sets itself around a 2 dimensional board, with a number of slots. In a turn, first player 1 gets to do a move, and subsequently player 2. In a move, a player chooses a column of the board where it will place its chip. The chip will be placed on the bottom of that column if there is not a chip in that particular slot yet. Otherwise, it will be placed on top of the already occupied slots in the respective column.

The game ends when either the board is full (a draw) or a victorious situation is reached by one of the players. The players can do so by connecting four chips in either vertical, horizontal or diagonal.

Literature

The expectation is that if fuzzy logic is applied to the discrete decision game, it will be able to win in a game against algorithms currently equipped useful intelligent tool to play the game.

2. Methods

2.1 System and tools

The system used was a HP EliteBook 8570w with an 2014 intel i7 processor. The agent was implemented in python 2.7.\*

2.2 Setup

2.2.1 The experiment

In this paper there was an experiment performed to play the game of Connect-4 with an AI based on a fuzzy-controller. To evaluate the performance of this fuzzy agent, the AI has been put in a competition against two other AI players. The first is one based on Monte Carlos search algorithm, the same algorithm that was produced in the famous alpha Go (Silver et al., 2016). The second opponent uses a brute-force based approach. By looking at the results of games from the fuzzy agent against these two opponent something can be said about the performance of the fuzzy agent. Because both the fuzzy agent and the brute force agent are deterministic we only need to play 2 games to evaluate them. For the matches against MonteCarlos the game has been played 80 of times.

**Opponents:**

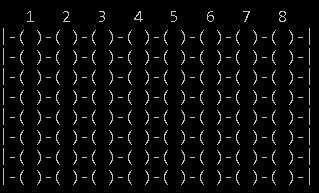
The variation on the Monte Carlos algorithm in this paper first collects all the possible moves at that moment in the game. Then, for each possible move it takes 0.8 seconds to evaluate that move. The evaluation goes as follows: it plays out as much games as it can within the given time limit, by just playing random moves. If it reaches an end situation it adds 1, 0 or -1 to a counter (won, tie, lose). This way it takes a random sample of the results of that move. The on average best scoring move gets chosen

The brute Force based algorithm also collects all the possible moves, and consequently it will play 3 moves and see how many of those moves lead to a victory in 1 turn, how many in 2 turns and how many in 3 turns. It then assigns a score to those numbers: 64,8,1 respectively. The best scoring move gets chosen.

From here we will refer to the fuzzy agent as the agent.

*\* Added for completeness. Since it is a subcategory of computational intelligence, performance is something that should be accounted for too.*

2.2.2 The board



*Figure 1:   
The playing board of the experiment*

Traditionally Connect-4 is played on a board with dimensions 7 times 6 (height and width respectively). This game however is already solved. The first player always wins if he/she plays perfectly. This among other reasons is because by playing the middle column, player 1 could always guarantee that player 2 would not catch a row-victory (Allis, 1988). This is why the playing field in this game is expanded to an even amount of columns and an higher amount of rows: 8x8.

2.3 What’s the Fuzz?

Unlike Boolean logic, fuzzy logic allows expressions and usage of vagueness and uncertainty. There is a variety of occasions where that is especially useful. Yet a game such as Connect-4 does not have a lot uncertainty in it, at least not in the conventional fuzzy logic way. It is a discrete game, based on a decision tree with a limited amount of choices. There is no column 6.4 or 3.2 and it would not be logical round them up or down because the output in columns is not continues nor linear. For example in a particular state in the game column 5 might be an excellent move, 6 a horrible one and 7 relatively good again. The score of one column does not imply anything about its neighboring column. If the columns would be the crisp output of the system that would a problem, because there is no clear range.

Instead, the agent will use fuzzy logic on evaluating the moves that are available. In fuzzy logics we can evaluate uncertainties we have about certain states of variables. Is the water out of the warm faucet hot? It is not true or false. It might be “slightly hot”, and “barely cold”. The same principle will be applied to the fuzzy agent. If I choose column 6 is my new state better, amazingly better or slightly worse (etc.) ? The linguistic approach will guide the agent through the vagueness of evaluating and choosing a move.

2.4 Variables

To evaluate however, we need a crisp input. The agent will do so by finding and then counting for certain patterns in the board. These are patterns based on the brute force agents evaluating system. The count of the following patterns will form the crisp input for the fuzzy controller:

2.4.1 input variables

Firstly, there is the “Potentials”. These are single moves with enough space around them to eventually make a connect-4. An example of a board(4x1) with potentials(board) = 1:



Secondly, there is the “Win-In-2’s”. This is the amount of possibilities that could lead to a victorious situation within 2 moves. An example of a board(5x1) with Win-In-2(board) = 2:



Thirdly, there is the “Win-In-1’s”. This is the amount of possibilities that could lead to a victorious situation within 1 move. An example of a board(1x4) with Win-In-1(board) = 1:



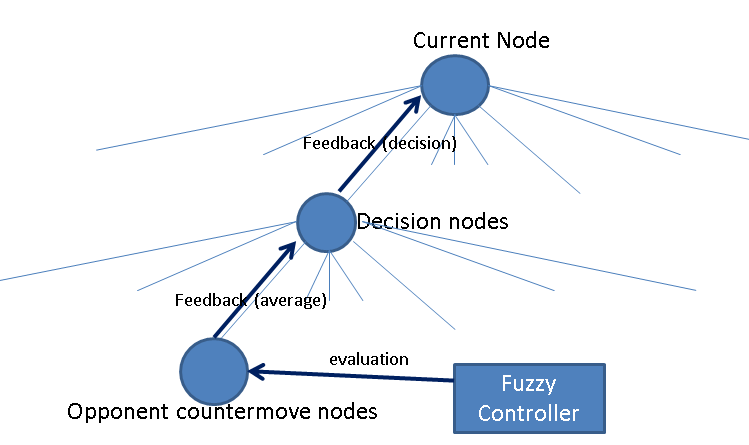
Finally, there is the “Progression”. This is the amount of turns that have been played so far. This feature is added because it is desirable that the agent strives for a victorious situation as soon as possible.

This last variable is only used as input once. The first three however, can be extended to a maximum of 3\*64=192 variables, because also the change in numbers of those variables can be taken into account so that the agent also strives for improvement. Using all 192 possible variables will most likely not help significantly to improve the performance of the agent, but in agent two and agent three 3\*3=9 of these are used. We will get to that later on (\*).

2.4.2 output variable

The output of the fuzzy logic controller is a grade between -1 and 11, describing the quality of the state of the game in which the evaluation took place.

*Figure 2: illustration of algorithm*

2.5 Algorithm

Initially the agent will collect its possible moves (decision nodes), then collect all the child-nodes in which the opponent makes a countermove. In these child-nodes the agent will use crisp input variables based on the evaluation method of the brute force agent as described above. Consequently the agent will take the average of the evaluation in the child-nodes (countermoves), and assign that average to the parent node (decision nodes) .

The fuzzy system is of a Mamdani type and uses the min-and method, the product-implication method and the centroid-deffuzification method.

Three different versions of the fuzzy agent have been implemented. One based on “expert knowledge” (fuzzy agent 1 : appendix B.1), one based on learning data from the brute force agent (fuzzy agent 2: appendix B.2) and the third (fuzzy agent 3 : appendix B.2) is also based on the same data but has an simple extra defense mechanism: if any of the counter moves of the opponent leads to a loss, don’t choose the move that leaded you there.

Agent 2 and 3 have as input variables progression and for the other variables they look 2 turns (meaning they also use the input variables of 1 turn ago and 2 turns ago) resulting in a total of 10 variables, whilst agent 1 takes in only progression, win-in-1 and win-in-2 in the evaluated move, resulting in 3 input variables\*. From this point on, the paper will be referring to agent 2 and 3 as the agent.

2.6 Learning

2.6.1 Training data

In order to make the agent more precise, data learning was applied. First the dataset was created out of the evaluation method of the brute force agent. This agent was chosen as an example because it performed relatively well against humans and it was easy to apply for the fuzzy agent since similar input variables are used. 500 games were played in which two random agents played each other, choosing a random move without trying to win every turn. The brute force agent would evaluate every move in between and assign it a score. The total score was then scaled between -1 and 11. This resulted in a dataset of 11,379 data points. Consequently the fuzzy agent could learn from it.

2.6.2 Learning memberships

To learn memberships each feature in the dataset was handled separately. First the data was analysed on how many clusters it could be identified on intuition. After that, the feature, the amount of estimated clusters and the scoring feature (output variable of the dataset) were used as inputs for the C-means (Dunn, 1973) clustering algorithm (m=2.6). The algorithm outputs the location of- and memberships of data points to the clusters in the feature-output space. We can the define a 3-dimensional result space: the feature-output-membership space. Next, we project all the output axis onto the feature-axis and that leaves us with a membership-feature space on which we can then fit a Gaussian membership function for each cluster. The standard deviations and centers of these Gaussians will define the membership functions for our input variables.

2.6.3 Learning Rules

After learning the memberships, the fuzzy rules can be determined. The agent that has been developed for this paper uses Wang and Mendels (1992) rule learning technique. After applying the cleanup of the rules as in step 3 and 4 in Wang and Mendel (1992) 127 of the 11,379 rules remain.

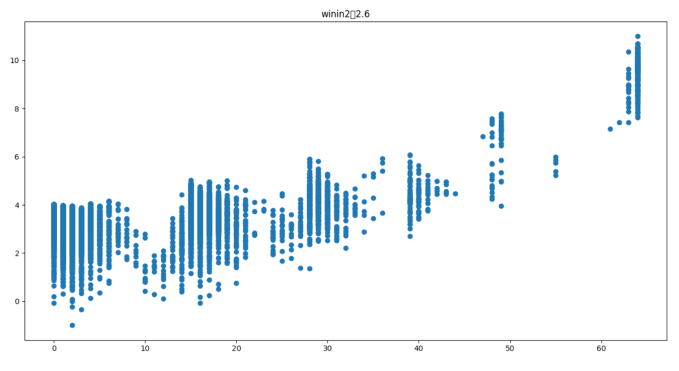
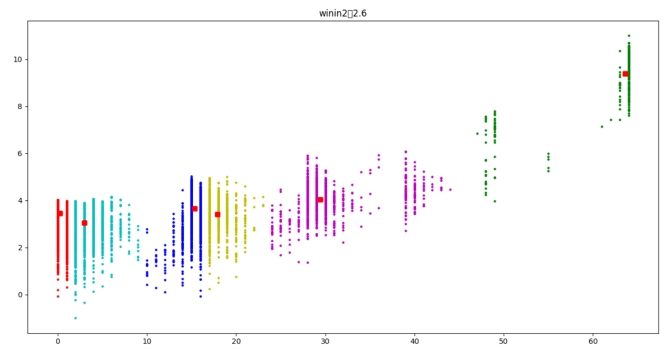
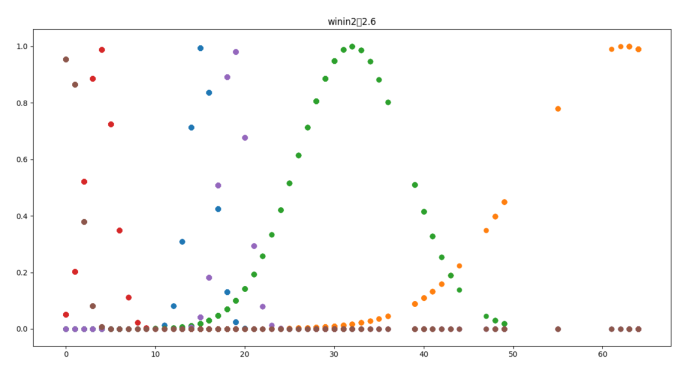
2.6.4 manual adjustment

Some manual adjustments have been done. Because with the amount of variables it seemed that no rule would ever have a firing strength over 1e-17, so the solution was to increase the standard deviation off all memberships with a factor 10 and add 10 as well. This is done after file reading, the membership parameters stated in appendix B are before manual adjustment.

3. Results

*Figure 3: Result of competition. Cells indicate the victories/losses for the fuzzy agent*

|  |  |  |
| --- | --- | --- |
| Fuzzy Agent starts | **Monte Carlos** | **Brute Force** |
| Fuzzy agent 1 | 8.75% | won |
| Fuzzy agent 2 | 0% | lost |
| Fuzzy agent 3 | 0% | lost |
| Opponent starts | **Monte Carlos** | **Brute Force** |
| Fuzzy agent 1 | 12.5% | won |
| Fuzzy agent 2 | 0% | lost |
| Fuzzy agent 3 | 3.75% | won |

  
As can be seen in figure 3; agent 1 outperforms the brute force agent and the other fuzzy agent. The agent based on Monte Carlos search algorithm is the best agent in the competition. Remarkably, all fuzzy agents score either worse or equally good when they start the game than when they do not.

Agent 2 is the worst performing version of the fuzzy agents. It seems to prefer the offensive, but does not care for defense. For example, when the opponent has three disks in a row and agent 2 has two, it prefers to go for own gain rather than blocking the opponent. Because of this seemingly selfish (and ignorant) behavior a simple defensive method needed to be added. Thus agent 3 was created, who performed slightly better.

3.1 Learning

All variables were trainable by the data. Most resulted in usable membership functions. Some had a few bad membership functions in them, which were removed manually.

Therefore rule generation also showed to be possible. A problem however was that because the large number of rules and variables, the firing strength stayed too low for the software to pick up on it. Hence the standard deviation of all membership functions needed to be raised in order for it to still work.

*Figure 4: Learning membership of win-in-2 variable (see other variables in appendix B) a) raw data, b) clustering result, c) result of learning membership functions*

3.2 Conclusion

The agents with rule learning performed worse than anything else (including agent 1), showing that either the specific dataset is not functional for this purpose or data learning is not the method for making an agent for a game with a discrete decision tree.

Furthermore, the system seems not to work with the trained data alone, manual adjustment needs to be done.

4. Discussion

There are multiple reasons why the fuzzy agents based on the dataset are not performing as well as expected. The first is that perhaps, even though the process of fitting memberships was succeeding, the dataset was not right for this AI. It is after all a dataset produced for the sole purpose of using it, and design biases could have been the reason. It could even be that certain situations in the dataset just do not happen in the real game, because the dataset is created by doing random moves. A new possible approach is to evaluate the fuzzy system on how many errors it makes to the perfect strategy of the game according VICTOR (Allis, 1988). The dataset could be evaluated with this scoring, and parameters of the fuzzy controller perfected with an evolutionary algorithm.

Moreover, the amount of variables can be the problem. Too many variables does not always mean better result. Sometimes too many variables make the coefficients too small and insignificant (Schneider & Wagemann 2010). This was clearly the case in this experiment, because a lot of rules did not fire.

Another possibility is that the way the data is used made the fuzzy agent not perform well. It became apparent that the firing strength of most rules was diminished too far due to the size of the antecedent. The fact that manual addition had to be done on the membership functions standard deviation is a warning for bad data handling. A more thorough examination on the rule- generating and exploitation should be done

However, research could be done to combining expert knowledge and data learning for a discrete problem such as connect-4. The rules the experts give could be combined in similar ways as step 4 in Wang and Mendel (1992). For combining knowledge and data into one system is fuzzy logic’s strength.

References

Allis, L. V. (1988). *A knowledge-based approach of connect-four*. Vrije Universiteit, Subfaculteit Wiskunde en Informatica.

Arfi, B. (2006). Linguistic fuzzy-logic game theory. *Journal of conflict resolution*, *50*(1), 28-57.

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters.

Pang, W., Wang, K. P., Zhou, C. G., & Dong, L. J. (2004, September). Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In *Computer and Information Technology, 2004. CIT'04. The Fourth International Conference on* (pp. 796-800). IEEE.

Pirovano, M., & Lanzi, P. L. (2014). Fuzzy Tactics: A scripting game that leverages fuzzy logic as an engaging game mechanic. *Expert Systems with Applications*, *41*(13), 6029-6038.

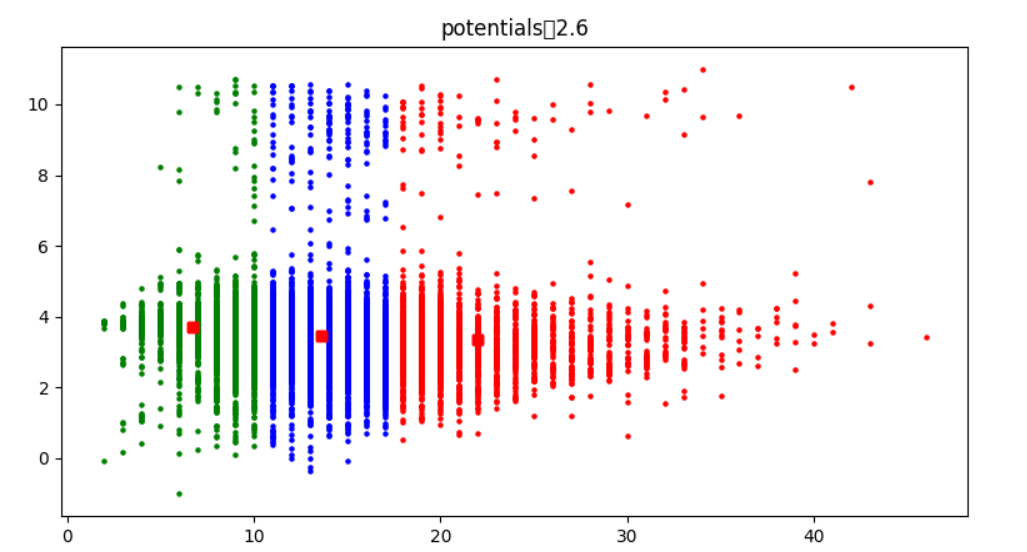
Sahin, A., & Kumbasar, T. (2017, July). Landing on the moon with type-2 fuzzy logic. In *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on* (pp. 1-6). IEEE.

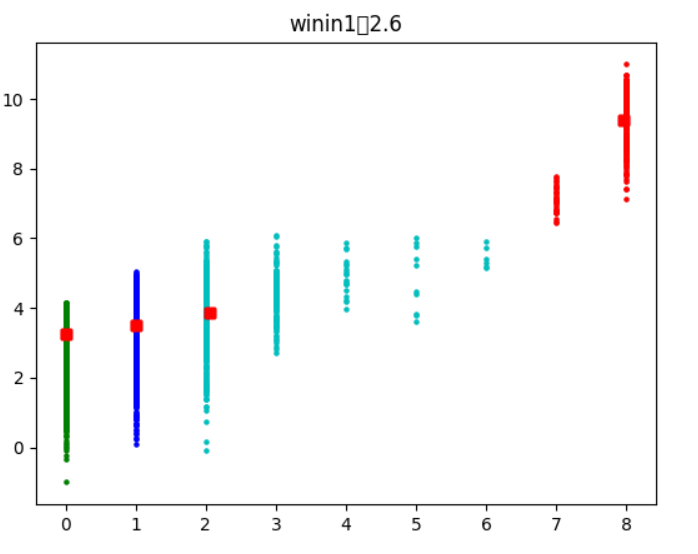
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484-489.

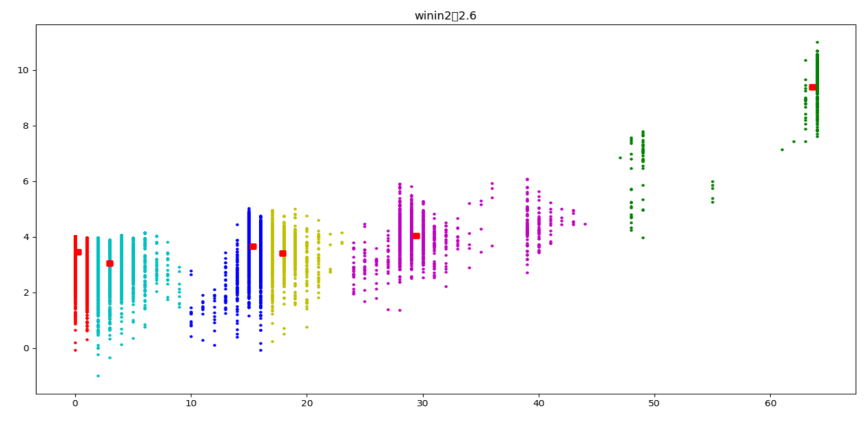
Schneider, C. Q., & Wagemann, C. (2010). Standards of good practice in qualitative comparative analysis (QCA) and fuzzy-sets. *Comparative Sociology*, *9*(3), 397-418.

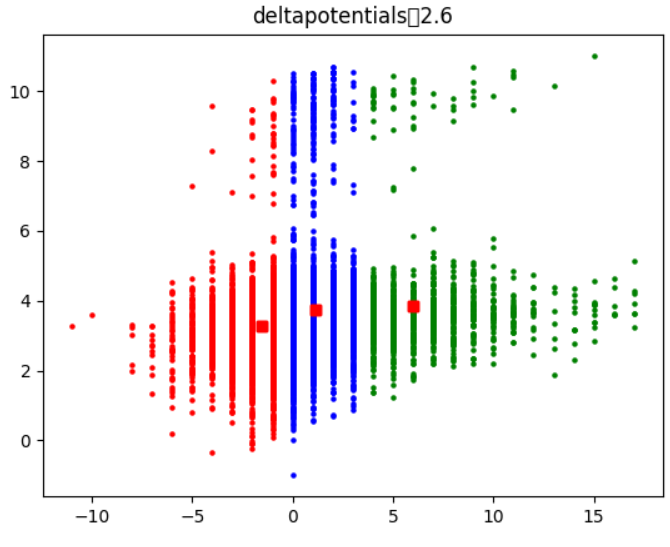
Wang, L. X., & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on systems, man, and cybernetics*, *22*(6), 1414-1427.

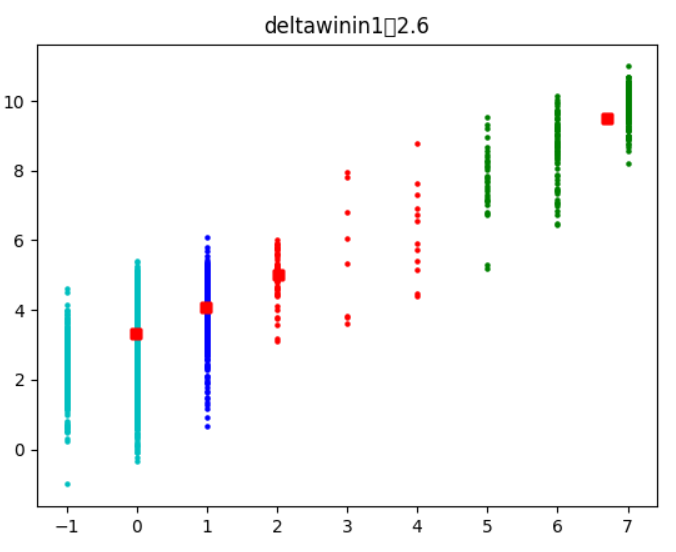
Appendix A

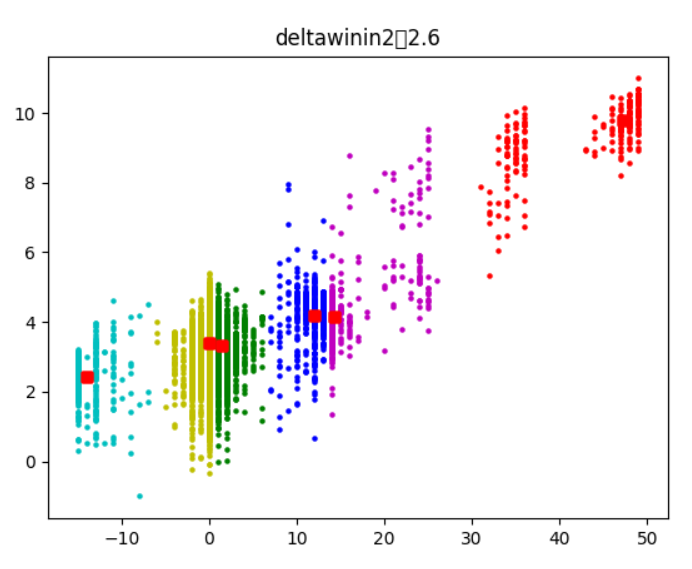


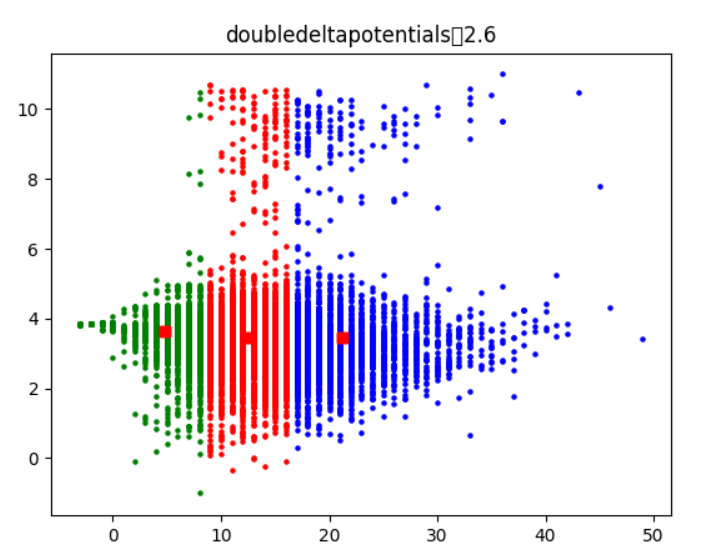


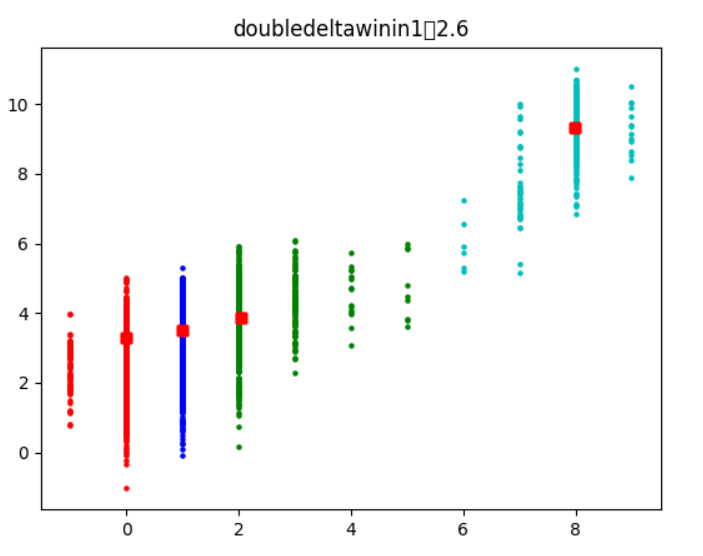


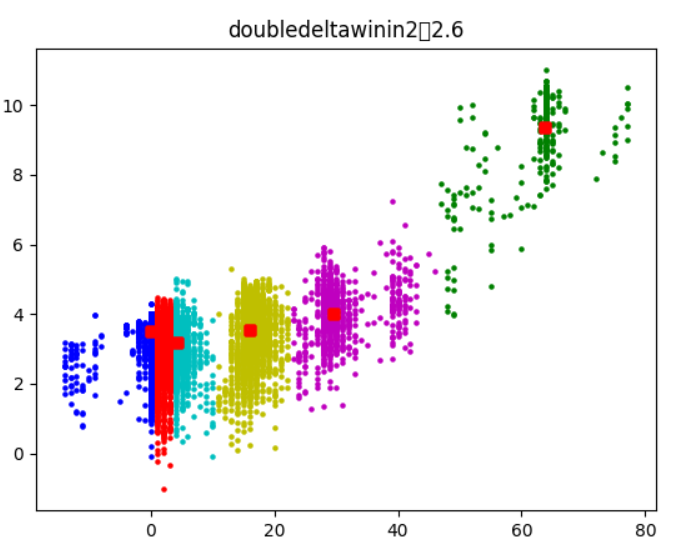


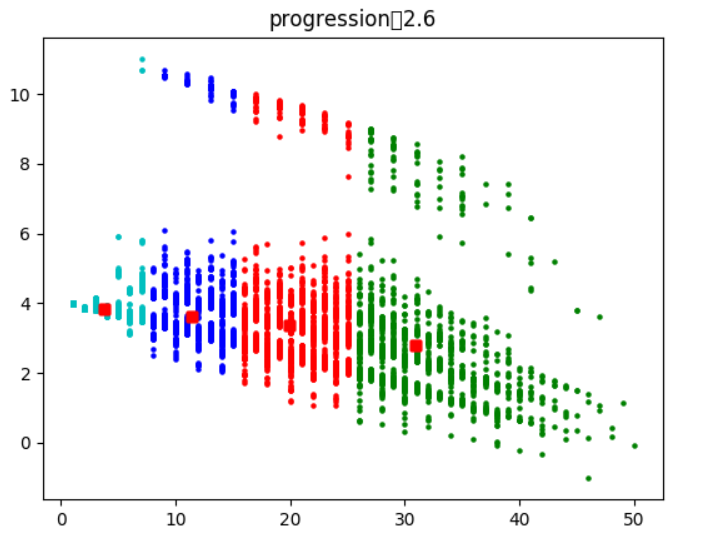


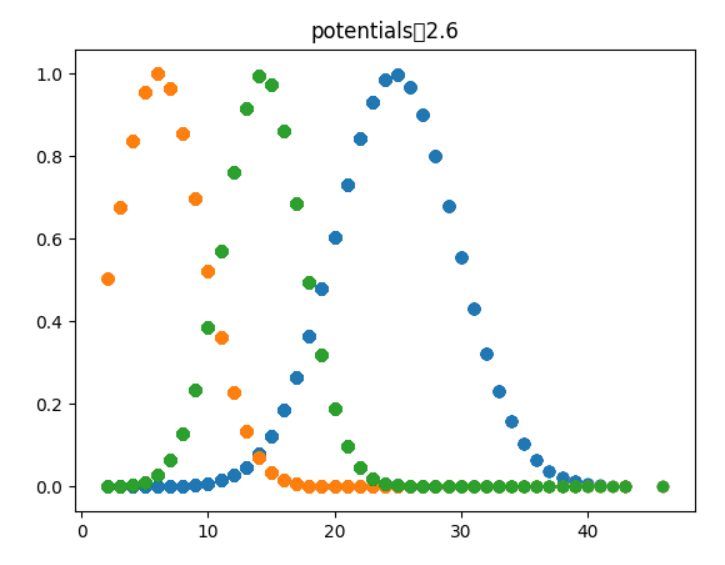


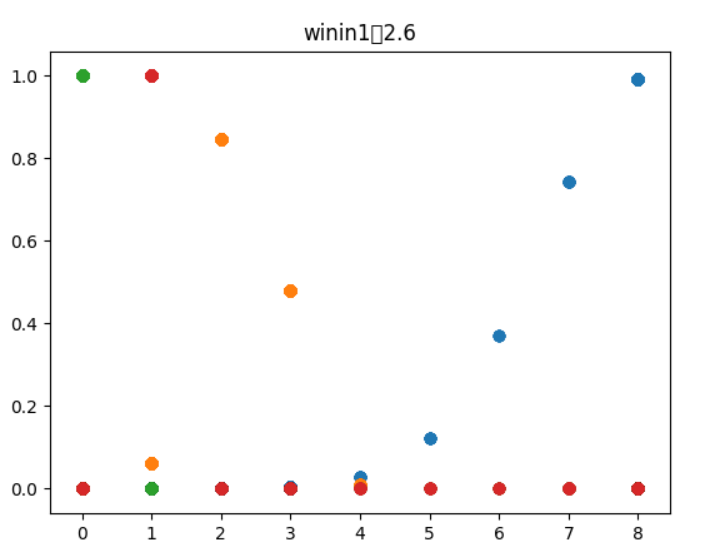


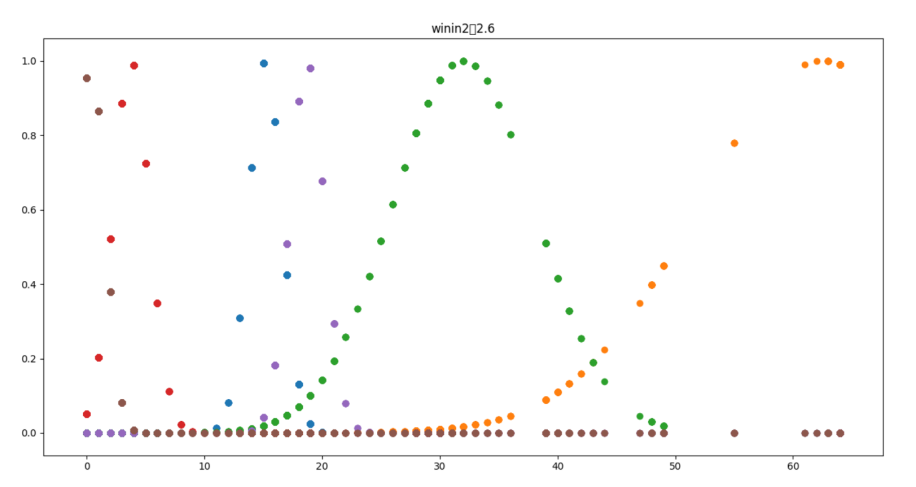


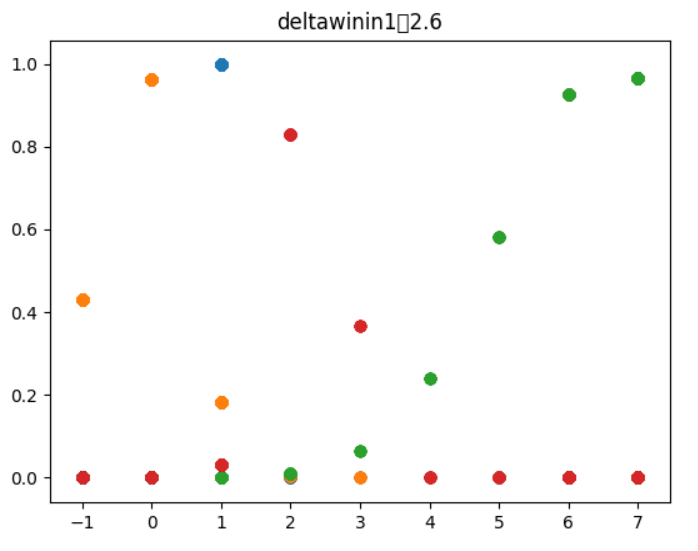
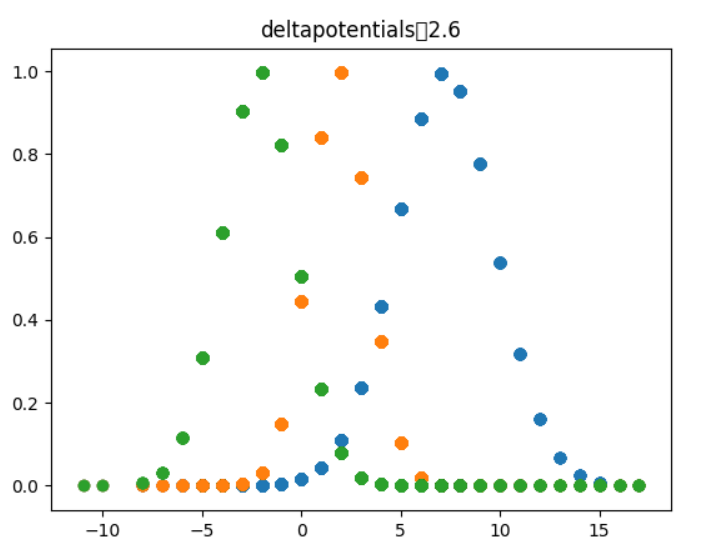


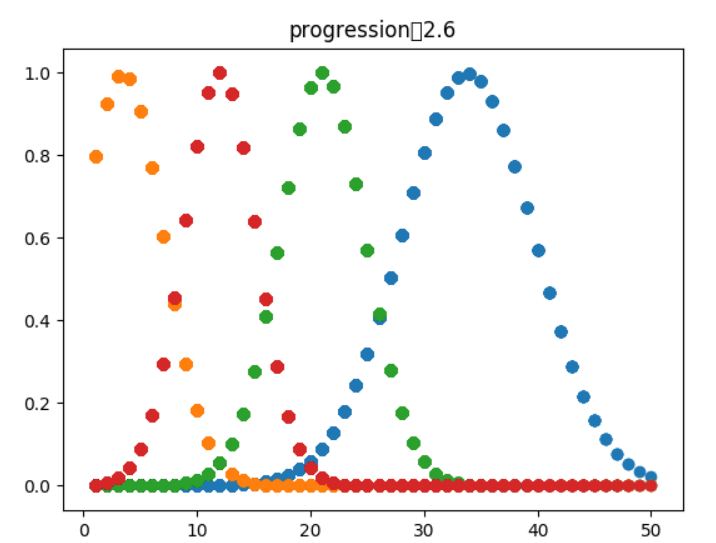
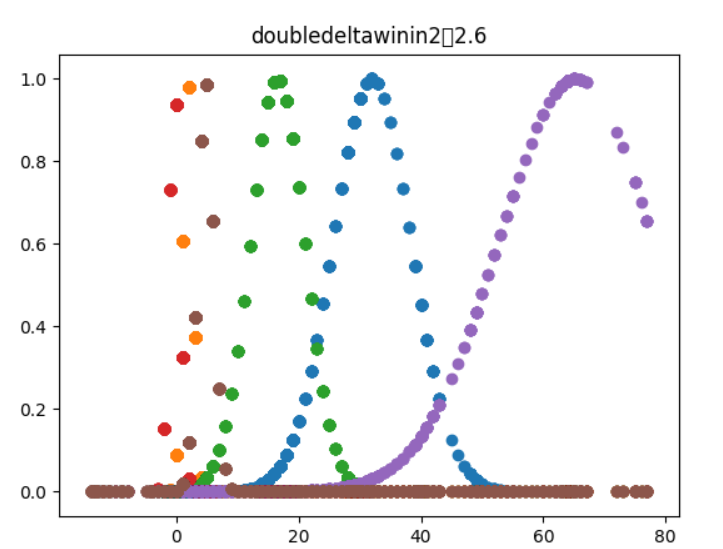
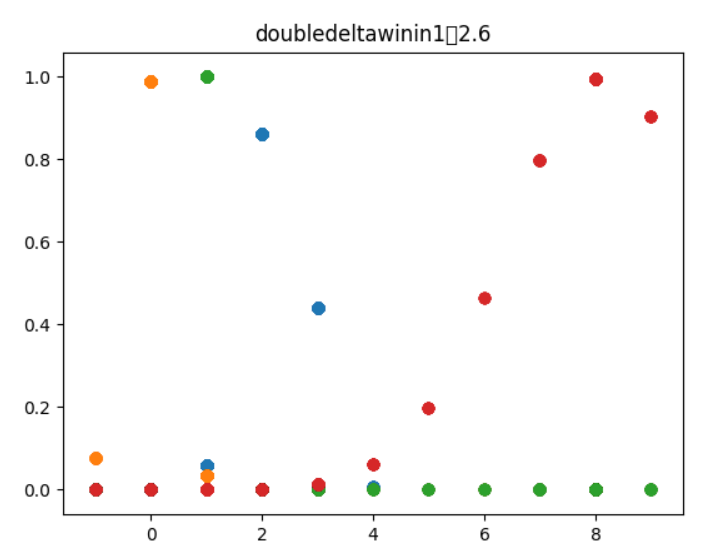
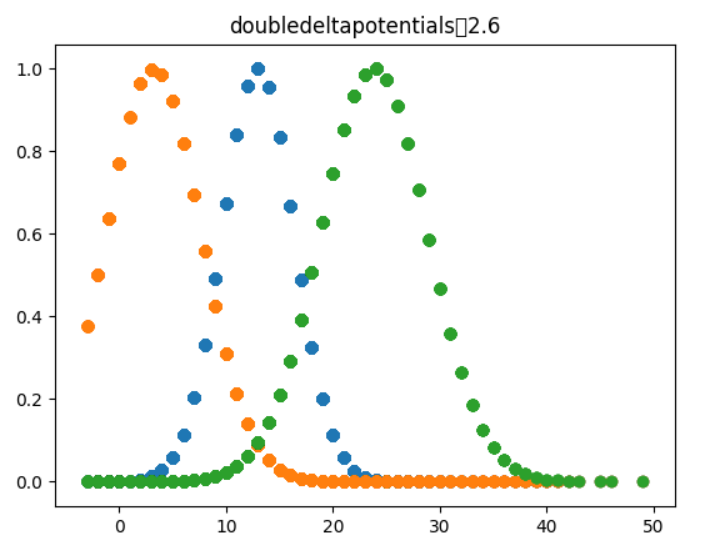
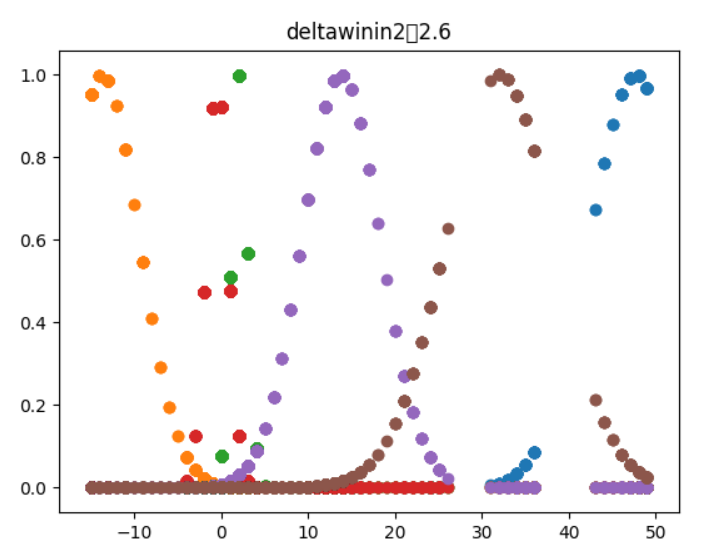












Appendix B

1 fuzzyAgent1 fis file

[System]

Name='LeafNodeSystem'

Type='mamdani'

Version=2.0

NumInputs=3

NumOutputs=1

NumRules=30

AndMethod='prod'

OrMethod='probor'

ImpMethod='prod'

AggMethod='sum'

DefuzzMethod='centroid'

[Input1]

Name='GameProgression'

Range=[0 64]

NumMFs=4

MF1='early':'trimf',[-25.6 0 30]

MF2='mid-late':'trimf',[10 42 72]

MF3='late':'trimf',[34 64 89.6]

MF4='mid-early':'trimf',[-10 20 48]

[Input2]

Name='winIn1'

Range=[0 8]

NumMFs=4

MF1='nothing':'trapmf',[0 0 1 1]

MF2='certainWin':'trapmf',[1 2 8 8]

MF3='won':'trapmf',[3 6 8 8]

MF4='good':'trapmf',[0 1 2 3]

[Input3]

Name='winIn2'

Range=[0 64]

NumMFs=4

MF1='few':'trimf',[-25.6 0 15]

MF2='some':'trapmf',[0 15 25 40]

MF3='won':'trapmf',[45 60 64 65]

MF4='lots':'trapmf',[15 30 60 64]

[Output1]

Name='ownGain'

Range=[-1 11]

NumMFs=11

MF1='absent':'trapmf',[-1 -1 0 1]

MF2='microscopic':'trapmf',[0 1 2 3]

MF3='tiny':'trapmf',[1 2 3 4]

MF4='slight':'trapmf',[2 3 4 5]

MF5='small':'trapmf',[3 4 5 6]

MF6='some':'trapmf',[4 5 6 7]

MF7='medium':'trapmf',[5 6 7 8]

MF8='reasonable':'trapmf',[6 7 8 9]

MF9='large':'trapmf',[7 8 9 10]

MF10='huge':'trapmf',[8 9 10 11]

MF11='won':'trapmf',[9 10 11 11]

[Rules]

0 2 0, 10 (1) : 1

0 3 0, 11 (1) : 1

1 4 0, 10 (1) : 1

4 4 0, 9 (1) : 1

2 4 0, 9 (1) : 1

3 4 0, 8 (1) : 1

1 1 0, 7 (1) : 1

4 1 0, 6 (1) : 1

2 1 0, 5 (1) : 1

3 1 0, 1 (1) : 1

0 0 3, 11 (1) : 1

1 0 1, 7 (1) : 1

4 0 1, 5 (1) : 1

2 0 1, 3 (1) : 1

3 0 1, 1 (1) : 1

1 0 2, 8 (1) : 1

4 0 2, 8 (1) : 1

2 0 2, 7 (1) : 1

3 0 2, 3 (1) : 1

1 0 4, 9 (1) : 1

4 0 4, 8 (1) : 1

2 0 4, 7 (1) : 1

3 0 4, 6 (1) : 1

0 1 1, 1 (1) : 1

0 1 2, 4 (1) : 1

0 1 4, 6 (1) : 1

0 4 1, 6 (1) : 1

0 4 2, 7 (1) : 1

0 4 4, 8 (1) : 1

0 2 0, 10 (1) : 1

1 3 3, 11 (1) : 1

3 1 1, 1 (1) : 1

2 fuzzyAgent2 and fuzzyAgent3 fis file

[System]

Name='LeafNodeSystem'

Type='mamdani'

Version=2.0

NumInputs=3

NumOutputs=1

NumRules=127

AndMethod='min'

OrMethod='probor'

ImpMethod='prod'

AggMethod='sum'

DefuzzMethod='centroid'

[Output1]

Name='ownGain'

Range=[-1 11]

NumMFs=11

MF1='absent':'trapmf',[-1 -1 0 1]

MF2='microscopic':'trapmf',[0 1 2 3]

MF3='tiny':'trapmf',[1 2 3 4]

MF4='slight':'trapmf',[2 3 4 5]

MF5='small':'trapmf',[3 4 5 6]

MF6='some':'trapmf',[4 5 6 7]

MF7='medium':'trapmf',[5 6 7 8]

MF8='reasonable':'trapmf',[6 7 8 9]

MF9='large':'trapmf',[7 8 9 10]

MF10='huge':'trapmf',[8 9 10 11]

MF11='won':'trapmf',[9 10 11 11]

[Input1]

Name='winin1'

Range=[0 8]

NumMFs=4

MF1='1|4':'gaussmf',[0.000305971185071 0.220034679968]

MF2='2|4':'gaussmf',[8.20400129296 1.56806970103]

MF3='3|4':'gaussmf',[1.01698628033 0.291200998922]

MF4='4|4':'gaussmf',[2.32113862034 0.55889661412]

[Input2]

Name='progression'

Range=[1 64]

NumMFs=4

MF1='1|4':'gaussmf',[0.40900298369 15.57906409119]

MF2='2|4':'gaussmf',[33.8130835241 5.82874647569]

MF3='3|4':'gaussmf',[21.0191560734 3.75449046552]

MF4='4|4':'gaussmf',[64.0 15.18721932131]

[Input3]

Name='doubledeltapotentials'

Range=[-3 49]

NumMFs=3

MF1='1|3':'gaussmf',[23.8306066411 5.00192230181]

MF2='2|3':'gaussmf',[3.20338635516 4.43473803282]

MF3='3|3':'gaussmf',[12.9794650339 3.35315969484]

[Input4]

Name='deltawinin1'

Range=[-1 7]

NumMFs=4

MF1='1|4':'gaussmf',[-0.17501532836 0.636080713338]

MF2='2|4':'gaussmf',[2.29514103644 0.490557467617]

MF3='3|4':'gaussmf',[1.01302920617 0.259181817953]

MF4='4|4':'gaussmf',[6.59816188961 1.54280344653]

[Input5]

Name='deltawinin2'

Range=[-15 49]

NumMFs=6

MF1='1|6':'gaussmf',[-13.6909034652 4.25962546222]

MF2='2|6':'gaussmf',[47.6463222646 5.24500135435]

MF3='3|6':'gaussmf',[32.0188415458 6.24057525174]

MF4='4|6':'gaussmf',[13.7874720384 4.46360668959]

MF5='5|6':'gaussmf',[-0.497138748543 1.22861375964]

MF6='6|6':'gaussmf',[2.04386280699 1.899921955219]

[Input6]

Name='potentials'

Range=[2 46]

NumMFs=3

MF1='1|3':'gaussmf',[14.2983782176 3.11755491284]

MF2='2|3':'gaussmf',[24.7993850847 4.79026461685]

MF3='3|3':'gaussmf',[6.05866162206 3.46511627362]

[Input7]

Name='deltapotentials'

Range=[-11 17]

NumMFs=3

MF1='1|3':'gaussmf',[1.89689635008 1.46973687873]

MF2='2|3':'gaussmf',[-2.15168161852 1.86198545694]

MF3='3|3':'gaussmf',[7.26000667241 2.49069171018]

[Input8]

Name='doubledeltawinin2'

Range=[-14 77]

NumMFs=6

MF1='1|6':'gaussmf',[-0.314241027842 0.881282907168]

MF2='2|6':'gaussmf',[4.77126679401 1.34275678115]

MF3='3|6':'gaussmf',[31.9942211436 6.36671900487]

MF4='4|6':'gaussmf',[1.8415518431 0.835468144209]

MF5='5|6':'gaussmf',[16.5212831949 4.44240419081]

MF6='6|6':'gaussmf',[65.354537477 12.6592926813]

[Input9]

Name='doubledeltawinin1'

Range=[-1 9]

NumMFs=4

MF1='1|4':'gaussmf',[8.19831160219 1.77523888453]

MF2='2|4':'gaussmf',[0.999694652069 0.0707425801551]

MF3='3|4':'gaussmf',[-0.0654699348918 0.410989681851]

MF4='4|4':'gaussmf',[2.30029309908 0.546563594716]

[Input10]

Name='winin2'

Range=[0 64]

NumMFs=6

MF1='1|6':'gaussmf',[16.4797586176 4.40359666466]

MF2='2|6':'gaussmf',[0.139635217424 1.738669344503]

MF3='3|6':'gaussmf',[62.3166865669 10.8543627245]

MF4='4|6':'gaussmf',[2.14415804747 1.86094707619]

MF5='5|6':'gaussmf',[31.952659344 6.37311179832]

MF6='6|6':'gaussmf',[4.81392262926 1.19960969653]

[Rules]

2 4 1 4 6 2 3 5 1 3, 3 (1) : 1

2 2 1 4 2 2 3 2 1 3, 4 (1) : 1

2 3 1 2 2 1 3 6 1 3, 4 (1) : 1

2 2 1 2 2 1 3 2 1 3, 4 (1) : 1

2 3 1 2 2 1 3 3 1 3, 5 (1) : 1

2 2 1 2 2 2 3 6 2 3, 4 (1) : 1

2 3 1 4 2 2 3 6 2 3, 4 (1) : 1

2 2 3 4 2 2 3 6 2 1, 4 (1) : 1

2 2 3 4 2 2 3 6 2 3, 3 (1) : 1

2 2 1 2 2 2 3 6 1 5, 4 (1) : 1

2 2 3 4 2 2 3 6 1 3, 4 (1) : 1

2 2 1 2 2 1 3 6 1 3, 3 (1) : 1

2 2 1 4 3 2 3 3 1 5, 11 (1) : 1

2 2 1 4 2 1 3 6 1 1, 4 (1) : 1

2 2 3 4 2 2 2 6 1 5, 4 (1) : 1

2 3 1 2 2 2 3 6 2 3, 2 (1) : 1

2 4 1 2 6 2 3 3 1 3, 4 (1) : 1

2 2 1 4 2 1 3 6 1 3, 4 (1) : 1

2 3 1 4 2 1 3 2 1 3, 2 (1) : 1

2 2 1 2 2 2 3 5 1 3, 4 (1) : 1

2 3 1 4 2 1 3 6 1 3, 3 (1) : 1

2 3 1 2 2 2 3 3 1 5, 6 (1) : 1

2 3 1 2 6 2 3 5 1 3, 4 (1) : 1

2 2 3 2 2 1 3 6 1 1, 4 (1) : 1

2 2 1 4 2 2 2 6 1 1, 4 (1) : 1

2 3 1 2 2 2 3 3 1 1, 6 (1) : 1

2 2 1 2 2 1 3 6 1 1, 4 (1) : 1

2 2 3 4 2 2 3 2 1 3, 3 (1) : 1

2 2 3 4 2 2 3 6 1 1, 4 (1) : 1

2 3 1 4 2 1 3 2 1 5, 5 (1) : 1

2 3 1 2 2 2 3 3 1 3, 2 (1) : 1

2 4 1 2 2 2 3 5 1 3, 5 (1) : 1

2 2 1 2 2 2 3 6 1 3, 4 (1) : 1

2 4 1 2 6 2 3 5 1 3, 5 (1) : 1

2 4 1 4 2 2 3 6 1 3, 5 (1) : 1

2 2 3 2 2 2 3 6 1 1, 4 (1) : 1

2 3 1 2 6 2 3 2 1 5, 5 (1) : 1

2 2 3 4 2 2 2 6 1 3, 4 (1) : 1

2 3 1 4 2 2 2 6 1 5, 5 (1) : 1

2 2 1 4 2 2 3 2 1 5, 4 (1) : 1

2 3 1 4 6 2 3 5 1 3, 3 (1) : 1

2 3 1 2 2 1 3 2 1 5, 5 (1) : 1

2 2 1 4 2 2 3 6 1 5, 4 (1) : 1

2 2 1 2 3 2 3 3 1 3, 7 (1) : 1

2 2 3 4 2 1 3 6 1 5, 4 (1) : 1

2 2 1 2 3 2 3 6 1 3, 10 (1) : 1

2 2 3 4 2 1 3 6 1 3, 4 (1) : 1

2 3 1 4 2 1 3 5 1 3, 4 (1) : 1

2 2 3 4 2 2 3 6 1 5, 4 (1) : 1

2 3 1 2 2 2 3 2 1 3, 2 (1) : 1

2 2 1 4 2 2 3 6 2 6, 4 (1) : 1

2 3 1 2 2 1 3 2 1 3, 5 (1) : 1

2 2 3 4 2 2 3 6 2 5, 4 (1) : 1

2 2 1 4 2 1 3 2 1 3, 3 (1) : 1

2 2 1 4 2 1 3 2 1 5, 4 (1) : 1

2 4 1 2 2 1 3 5 1 3, 4 (1) : 1

2 2 3 2 2 2 3 6 1 5, 4 (1) : 1

2 2 1 2 1 2 3 3 1 3, 6 (1) : 1

2 2 1 4 1 2 3 3 1 3, 6 (1) : 1

2 2 1 2 2 2 2 6 2 5, 4 (1) : 1

2 2 3 2 2 1 3 6 1 5, 4 (1) : 1

2 2 3 4 2 1 3 6 1 1, 4 (1) : 1

2 2 1 2 2 2 3 2 1 3, 4 (1) : 1

2 3 1 4 6 2 3 2 1 3, 2 (1) : 1

2 2 1 4 2 2 3 6 1 3, 4 (1) : 1

2 3 1 2 2 2 3 6 2 5, 5 (1) : 1

2 3 1 4 2 2 3 6 1 3, 3 (1) : 1

2 4 1 2 2 2 3 3 1 3, 4 (1) : 1

2 3 1 2 6 2 3 2 1 3, 2 (1) : 1

2 2 3 2 2 2 3 6 2 5, 4 (1) : 1

2 3 1 4 2 2 3 3 1 3, 2 (1) : 1

2 2 1 4 2 2 2 6 1 5, 4 (1) : 1

2 2 1 4 2 2 2 6 2 5, 4 (1) : 1

2 4 1 4 2 2 3 2 1 3, 5 (1) : 1

2 2 3 2 2 2 3 6 2 3, 4 (1) : 1

2 2 1 2 2 2 3 3 1 3, 6 (1) : 1

2 2 1 4 2 2 2 6 2 1, 4 (1) : 1

2 4 1 4 2 2 3 3 1 3, 4 (1) : 1

2 3 1 4 2 2 3 6 2 5, 5 (1) : 1

2 2 1 4 2 2 3 5 1 3, 4 (1) : 1

2 2 1 4 2 2 2 6 2 3, 4 (1) : 1

2 3 1 4 6 2 3 2 1 1, 5 (1) : 1

2 2 3 4 2 2 2 6 2 1, 4 (1) : 1

2 2 1 4 3 2 3 6 1 3, 8 (1) : 1

2 2 1 4 2 2 3 6 2 5, 4 (1) : 1

2 2 1 2 2 2 2 6 2 1, 4 (1) : 1

2 3 1 2 2 2 3 6 1 5, 5 (1) : 1

2 2 3 4 2 2 3 2 1 5, 4 (1) : 1

2 2 1 4 3 2 3 3 1 3, 7 (1) : 1

2 3 1 4 2 2 3 3 1 5, 5 (1) : 1

2 3 1 4 2 2 3 6 1 5, 5 (1) : 1

2 2 3 4 2 2 2 6 1 1, 4 (1) : 1

2 2 1 2 2 2 3 6 2 5, 4 (1) : 1

2 3 1 2 6 2 3 3 1 3, 2 (1) : 1

2 3 1 2 2 2 3 2 1 5, 5 (1) : 1

2 4 1 4 6 2 3 3 1 3, 4 (1) : 1

2 2 1 2 2 2 3 6 1 1, 5 (1) : 1

2 4 1 2 2 1 3 3 1 3, 5 (1) : 1

2 2 1 4 1 2 3 5 1 3, 5 (1) : 1

2 2 3 4 2 1 3 2 1 3, 4 (1) : 1

2 2 1 2 2 1 3 6 1 5, 4 (1) : 1

2 4 1 4 2 2 3 5 1 3, 4 (1) : 1

2 4 1 4 2 1 3 3 1 3, 3 (1) : 1

2 3 1 4 2 1 3 3 1 3, 5 (1) : 1

2 3 1 4 2 2 3 2 1 3, 2 (1) : 1

2 4 1 4 6 2 3 2 1 3, 6 (1) : 1

2 3 1 4 6 2 3 3 1 3, 2 (1) : 1

2 4 1 4 2 1 3 5 1 3, 5 (1) : 1

2 3 1 4 2 2 2 2 1 3, 1 (1) : 1

2 2 1 4 2 1 3 6 1 5, 4 (1) : 1

2 2 3 2 2 1 3 6 1 3, 3 (1) : 1

2 3 1 4 2 2 3 2 1 5, 5 (1) : 1

2 2 1 4 2 2 3 6 2 3, 3 (1) : 1

2 2 3 2 2 2 3 6 2 1, 4 (1) : 1

2 2 1 4 2 2 3 6 1 1, 5 (1) : 1

2 4 1 2 2 2 3 2 1 3, 5 (1) : 1

2 2 3 2 2 2 3 2 1 3, 3 (1) : 1

2 2 3 2 2 2 3 6 1 3, 3 (1) : 1

2 3 1 4 6 2 3 2 1 5, 5 (1) : 1

2 3 1 2 6 2 3 2 1 1, 5 (1) : 1

2 3 1 2 2 2 3 6 1 3, 4 (1) : 1

2 2 1 4 2 2 3 6 2 1, 4 (1) : 1

2 3 1 2 2 2 3 5 1 3, 5 (1) : 1

2 2 3 2 2 1 3 2 1 3, 4 (1) : 1

2 2 1 4 2 2 3 3 1 3, 6 (1) : 1

2 3 1 4 2 2 3 5 1 3, 3 (1) : 1

2 2 1 2 2 2 3 6 2 1, 4 (1) : 1

2 2 1 2 2 2 2 6 1 5, 4 (1) : 1

2 3 1 2 6 2 3 3 1 5, 6 (1) : 1

2 3 1 2 2 2 3 2 1 1, 5 (1) : 1

2 2 1 4 2 2 3 6 2 4, 4 (1) : 1

2 3 1 4 6 2 3 5 1 3, 4 (1) : 1

2 3 1 2 6 2 3 3 1 3, 6 (1) : 1

2 3 1 4 6 2 3 3 1 5, 6 (1) : 1

2 3 1 2 6 2 3 3 1 1, 6 (1) : 1

2 3 1 4 2 2 3 2 1 1, 5 (1) : 1

2 2 1 2 2 2 3 6 2 4, 4 (1) : 1

2 2 1 2 6 2 3 5 1 3, 6 (1) : 1

2 2 1 4 6 2 3 5 1 3, 7 (1) : 1

2 3 1 2 2 1 3 5 1 3, 4 (1) : 1

2 3 1 4 2 1 3 3 1 3, 3 (1) : 1

2 3 1 4 6 2 3 3 1 3, 6 (1) : 1

2 3 3 2 2 1 3 4 2 3, 3 (1) : 1

2 3 3 3 2 1 3 4 4 1, 4 (1) : 1

2 3 3 3 2 1 1 4 2 1, 4 (1) : 1

2 3 3 2 2 1 3 4 1 1, 4 (1) : 1

2 3 3 4 2 2 3 4 2 3, 3 (1) : 1

2 3 1 3 2 2 3 5 1 3, 5 (1) : 1

2 3 3 2 2 1 3 4 1 3, 4 (1) : 1

2 3 1 4 2 1 3 4 1 1, 4 (1) : 1

2 3 1 2 2 2 3 4 2 3, 4 (1) : 1

2 3 1 2 2 1 3 4 4 3, 5 (1) : 1

2 3 3 4 2 1 3 4 1 1, 4 (1) : 1

2 3 1 4 5 2 3 1 1 1, 5 (1) : 1

2 3 1 2 2 2 3 4 1 5, 4 (1) : 1

2 3 1 2 5 2 3 5 1 3, 5 (1) : 1

2 3 3 3 2 2 3 4 2 5, 4 (1) : 1

2 3 3 3 2 2 3 4 1 5, 4 (1) : 1

2 3 1 4 2 2 2 1 1 3, 4 (1) : 1

2 3 3 3 2 2 3 4 1 1, 5 (1) : 1

2 3 1 3 5 2 3 1 1 3, 2 (1) : 1

2 3 1 4 2 1 1 1 1 3, 5 (1) : 1

2 3 3 3 2 2 3 4 1 3, 4 (1) : 1

2 3 3 4 2 1 1 4 4 3, 4 (1) : 1

2 3 3 2 2 1 3 4 4 5, 4 (1) : 1

2 3 3 4 2 2 1 4 4 3, 3 (1) : 1

2 3 3 4 2 1 3 4 2 3, 3 (1) : 1

2 3 3 3 2 1 1 4 2 4, 4 (1) : 1

2 3 3 3 2 1 3 4 4 5, 4 (1) : 1

2 3 1 4 2 1 1 1 1 5, 5 (1) : 1

2 3 3 4 2 1 2 4 1 3, 4 (1) : 1

2 3 3 4 2 1 2 4 2 5, 4 (1) : 1

2 3 3 3 2 1 2 4 2 1, 4 (1) : 1

2 3 3 4 2 2 3 1 1 3, 2 (1) : 1

2 3 1 4 2 1 3 4 1 5, 5 (1) : 1

2 3 3 3 2 2 2 4 2 3, 2 (1) : 1

2 3 3 2 2 1 1 4 1 5, 4 (1) : 1

2 3 3 4 2 1 3 4 2 6, 4 (1) : 1

2 3 3 4 2 2 3 4 2 5, 4 (1) : 1

2 3 3 3 2 1 3 4 1 3, 3 (1) : 1

2 2 1 3 3 2 3 6 1 3, 10 (1) : 1

2 3 1 3 2 1 2 4 1 5, 5 (1) : 1

2 3 3 2 2 2 3 4 2 1, 4 (1) : 1

2 3 3 4 2 1 3 4 4 3, 3 (1) : 1

2 3 3 4 2 1 2 4 2 1, 4 (1) : 1

2 3 3 4 2 2 3 4 1 1, 5 (1) : 1

2 3 1 3 6 2 3 2 1 3, 4 (1) : 1

2 3 3 2 2 2 3 4 2 5, 4 (1) : 1

2 3 3 2 2 1 2 4 1 3, 3 (1) : 1

2 3 3 4 2 1 1 4 1 5, 4 (1) : 1

2 3 3 2 2 2 3 4 1 1, 4 (1) : 1

2 3 3 3 2 1 3 4 1 1, 4 (1) : 1

2 3 3 3 2 1 2 4 2 5, 4 (1) : 1

2 3 1 4 6 2 3 5 1 3, 6 (1) : 1

2 3 3 4 2 1 2 4 2 3, 4 (1) : 1

2 3 1 2 2 1 3 4 4 5, 5 (1) : 1

2 3 3 3 2 1 3 4 2 3, 3 (1) : 1

2 3 3 2 2 2 3 4 4 5, 4 (1) : 1

2 2 1 2 3 2 3 3 1 3, 9 (1) : 1

2 3 1 3 6 2 3 5 1 3, 4 (1) : 1

2 3 1 2 2 1 2 4 2 1, 4 (1) : 1

2 2 1 3 6 2 3 5 1 3, 6 (1) : 1

2 3 3 4 2 1 1 4 2 1, 4 (1) : 1

2 3 1 3 2 1 3 4 2 3, 4 (1) : 1

2 3 3 4 2 1 1 4 4 5, 4 (1) : 1

2 3 3 2 2 2 3 4 1 3, 4 (1) : 1

2 3 3 4 2 1 3 4 4 5, 4 (1) : 1

2 3 3 3 2 1 3 4 2 5, 4 (1) : 1

2 3 1 3 2 2 3 2 1 3, 2 (1) : 1

2 3 1 2 2 2 3 2 1 3, 4 (1) : 1

2 3 1 4 2 1 2 4 1 5, 5 (1) : 1

2 3 1 2 2 1 3 2 1 3, 2 (1) : 1

2 3 3 4 2 1 3 4 1 3, 3 (1) : 1

2 3 1 3 6 2 3 2 1 5, 6 (1) : 1

2 3 1 4 2 1 3 4 4 5, 5 (1) : 1

2 2 1 3 2 2 3 3 1 3, 6 (1) : 1

2 3 3 2 2 1 3 4 2 5, 4 (1) : 1

2 3 1 3 2 1 3 4 1 5, 4 (1) : 1

2 3 3 3 2 2 1 4 2 1, 4 (1) : 1

2 3 3 2 2 2 3 4 1 5, 4 (1) : 1

2 2 1 4 4 2 3 3 1 3, 6 (1) : 1

2 3 1 3 2 1 3 4 4 3, 3 (1) : 1

2 3 3 2 2 1 3 4 2 4, 4 (1) : 1

2 3 1 4 2 1 3 1 1 5, 5 (1) : 1

2 3 1 2 2 1 1 1 1 3, 3 (1) : 1

2 3 1 2 2 1 3 4 2 3, 4 (1) : 1

2 3 1 3 5 2 3 1 1 1, 5 (1) : 1

2 3 3 4 2 1 3 4 2 5, 4 (1) : 1

2 3 1 3 2 2 3 4 2 5, 5 (1) : 1

2 3 1 4 2 1 3 4 4 3, 4 (1) : 1

2 3 1 3 2 1 3 4 4 5, 5 (1) : 1

2 3 1 3 2 2 3 4 4 3, 4 (1) : 1

2 3 1 4 6 2 3 2 1 3, 6 (1) : 1

2 2 1 4 2 2 3 6 1 3, 9 (1) : 1

2 3 3 2 2 1 2 4 1 5, 4 (1) : 1

2 2 1 3 3 2 3 3 1 3, 7 (1) : 1

2 3 1 2 2 2 3 4 4 5, 5 (1) : 1

2 3 1 3 2 2 3 4 1 5, 4 (1) : 1

2 3 1 2 2 1 3 1 1 5, 5 (1) : 1

2 3 1 4 2 2 3 4 4 3, 3 (1) : 1

2 3 3 3 2 2 2 4 1 5, 4 (1) : 1

2 3 1 2 6 2 3 2 1 3, 5 (1) : 1

2 3 1 4 2 2 3 4 1 1, 5 (1) : 1

2 3 1 3 2 1 3 5 1 3, 3 (1) : 1

2 3 3 4 2 2 3 4 4 5, 4 (1) : 1

2 3 1 3 2 1 3 1 1 3, 2 (1) : 1

2 3 1 2 2 1 2 1 1 3, 1 (1) : 1

2 2 1 3 2 1 3 5 1 3, 6 (1) : 1

2 3 1 4 2 2 3 4 2 3, 4 (1) : 1

2 3 3 2 2 1 2 4 2 5, 4 (1) : 1

2 2 1 2 2 2 3 3 1 3, 8 (1) : 1

2 3 1 2 2 2 3 1 1 3, 2 (1) : 1

2 3 3 3 2 1 3 4 1 5, 4 (1) : 1

2 3 3 3 2 2 3 4 4 5, 4 (1) : 1

2 3 3 4 2 1 3 1 1 5, 4 (1) : 1

2 3 1 2 2 1 2 1 1 5, 5 (1) : 1

2 3 1 4 2 2 3 4 1 3, 4 (1) : 1

2 3 3 3 2 2 2 4 2 1, 4 (1) : 1

2 3 1 3 2 1 3 4 1 3, 3 (1) : 1

2 3 1 3 5 2 3 1 1 5, 5 (1) : 1

2 3 3 4 2 2 3 4 1 3, 2 (1) : 1

2 3 1 3 2 1 3 1 1 5, 5 (1) : 1

2 3 1 4 2 1 3 4 1 3, 4 (1) : 1

2 3 1 4 5 2 3 5 1 3, 4 (1) : 1

2 3 1 2 6 2 3 5 1 3, 6 (1) : 1

2 3 3 3 2 1 2 4 1 5, 4 (1) : 1

2 3 3 3 2 2 3 4 4 3, 4 (1) : 1

2 3 3 4 2 1 3 4 1 5, 4 (1) : 1

2 3 3 2 2 2 2 4 2 1, 4 (1) : 1

2 3 1 3 2 2 3 4 2 3, 2 (1) : 1

2 3 3 4 2 2 3 4 4 3, 3 (1) : 1

2 3 1 2 2 1 3 1 1 3, 2 (1) : 1

2 3 1 2 5 2 3 1 1 3, 5 (1) : 1

2 3 3 3 2 1 3 1 1 3, 4 (1) : 1

2 3 1 3 2 2 3 4 1 1, 5 (1) : 1

2 3 1 2 2 2 2 1 1 3, 4 (1) : 1

2 3 3 3 2 2 3 4 2 1, 4 (1) : 1

2 3 3 2 2 1 2 4 2 1, 4 (1) : 1

2 3 3 3 2 1 3 4 4 3, 3 (1) : 1

2 3 1 2 2 2 3 4 4 3, 4 (1) : 1

2 3 3 2 2 2 3 4 4 3, 4 (1) : 1

2 3 3 4 2 2 3 4 1 5, 4 (1) : 1

2 3 1 4 2 1 2 1 1 5, 5 (1) : 1

2 3 3 4 2 1 3 1 1 3, 4 (1) : 1

2 3 1 3 2 1 3 2 1 3, 2 (1) : 1

2 3 1 3 2 1 3 4 2 1, 4 (1) : 1

2 3 1 2 2 1 3 5 1 3, 5 (1) : 1

2 3 1 2 2 2 3 1 1 5, 5 (1) : 1

2 3 1 2 2 1 3 4 1 5, 5 (1) : 1

2 3 1 3 2 2 3 1 1 3, 2 (1) : 1

2 3 1 2 2 2 3 4 1 3, 3 (1) : 1

2 3 3 2 2 1 3 4 2 1, 4 (1) : 1

2 3 1 4 2 1 1 4 1 5, 5 (1) : 1

2 3 3 3 2 1 3 4 2 4, 4 (1) : 1

2 3 1 3 2 2 3 4 1 3, 4 (1) : 1

2 3 3 3 2 2 3 1 1 3, 3 (1) : 1

2 3 3 4 2 1 3 4 2 1, 4 (1) : 1

2 3 1 3 2 1 1 4 1 5, 4 (1) : 1

2 3 3 4 2 1 3 4 2 4, 4 (1) : 1

2 3 1 2 2 1 3 4 2 1, 4 (1) : 1

2 3 1 3 5 2 3 2 1 3, 2 (1) : 1

2 3 1 4 2 2 3 1 1 3, 5 (1) : 1

2 3 1 3 5 2 3 5 1 3, 3 (1) : 1

2 3 3 3 2 2 3 4 2 3, 4 (1) : 1

2 3 1 2 2 2 3 4 1 1, 4 (1) : 1

2 3 1 3 2 1 3 4 2 5, 4 (1) : 1

2 3 3 2 2 2 3 4 2 3, 4 (1) : 1

2 3 3 2 2 1 3 1 1 3, 4 (1) : 1

2 3 3 2 2 2 1 4 2 1, 4 (1) : 1

2 3 3 4 2 1 1 4 2 5, 4 (1) : 1

2 3 3 2 2 1 2 4 4 3, 4 (1) : 1

2 3 3 4 2 1 3 4 4 1, 4 (1) : 1

2 3 3 2 2 1 3 4 4 3, 4 (1) : 1

2 3 1 3 6 2 3 2 1 1, 6 (1) : 1

2 3 1 4 6 2 3 2 1 5, 6 (1) : 1

2 2 1 4 2 1 3 5 1 3, 6 (1) : 1

2 2 1 3 2 2 3 5 1 3, 4 (1) : 1

2 3 1 4 5 2 3 1 1 3, 4 (1) : 1

2 3 3 3 2 1 3 4 2 1, 4 (1) : 1

2 3 3 2 2 1 3 4 1 5, 4 (1) : 1

2 3 1 3 2 2 3 1 1 5, 5 (1) : 1

2 3 1 4 5 2 3 1 1 5, 5 (1) : 1

2 3 1 4 5 2 3 2 1 3, 2 (1) : 1

2 3 1 4 2 1 2 1 1 3, 4 (1) : 1

2 3 1 2 5 2 3 2 1 3, 3 (1) : 1

2 3 3 4 2 1 2 4 1 5, 4 (1) : 1

2 3 1 4 2 2 3 5 1 3, 5 (1) : 1

2 3 3 2 2 1 1 4 2 1, 4 (1) : 1

2 3 1 2 2 1 3 4 1 3, 3 (1) : 1

2 2 1 4 4 2 3 5 1 3, 5 (1) : 1

2 2 1 3 4 2 3 3 1 3, 6 (1) : 1

2 3 1 4 2 1 3 1 1 3, 3 (1) : 1

2 2 1 4 2 2 1 6 2 1, 4 (1) : 1

2 2 1 4 2 2 1 6 1 1, 4 (1) : 1

2 2 1 3 2 2 3 2 1 3, 4 (1) : 1

2 3 1 4 2 2 1 2 1 3, 3 (1) : 1

2 2 3 3 2 2 3 6 2 3, 4 (1) : 1

2 2 3 3 2 1 3 6 1 1, 4 (1) : 1

2 3 1 3 2 2 3 6 1 3, 4 (1) : 1

2 2 1 3 2 2 3 6 2 1, 4 (1) : 1

2 2 1 4 2 2 1 6 2 3, 4 (1) : 1

2 2 3 3 2 1 3 2 1 3, 4 (1) : 1

2 2 3 4 2 2 1 6 1 3, 4 (1) : 1

2 3 1 3 2 1 3 2 1 5, 5 (1) : 1

2 2 1 3 1 2 3 3 1 3, 6 (1) : 1

2 2 1 3 2 1 3 6 1 5, 4 (1) : 1

2 2 3 3 2 2 3 6 1 3, 3 (1) : 1

2 3 1 3 2 1 3 6 1 3, 4 (1) : 1

2 2 1 3 2 2 3 6 2 5, 4 (1) : 1

2 2 1 3 2 1 3 6 1 1, 4 (1) : 1

2 2 1 3 2 1 3 6 1 3, 3 (1) : 1

2 3 1 3 6 2 3 2 1 3, 6 (1) : 1

2 3 1 3 2 2 3 6 1 5, 5 (1) : 1

2 2 1 4 2 2 1 6 2 5, 4 (1) : 1

2 2 3 3 2 2 3 6 1 1, 4 (1) : 1

2 3 1 4 2 2 1 6 1 3, 5 (1) : 1

2 3 1 3 2 2 3 6 2 5, 5 (1) : 1

2 2 3 3 2 1 3 6 1 3, 3 (1) : 1

2 2 1 3 2 2 3 6 2 3, 4 (1) : 1

2 3 1 3 2 2 3 6 2 3, 2 (1) : 1

2 2 1 3 2 2 3 6 2 4, 4 (1) : 1

2 3 1 3 2 1 3 2 1 3, 5 (1) : 1

2 3 1 3 2 1 3 5 1 3, 4 (1) : 1

2 2 3 3 2 2 3 6 2 5, 4 (1) : 1

2 2 3 3 2 2 3 2 1 3, 3 (1) : 1

2 2 3 3 2 1 3 6 1 5, 4 (1) : 1

2 3 1 3 2 2 3 2 1 5, 5 (1) : 1

2 2 3 3 2 2 3 6 1 5, 4 (1) : 1

2 2 3 4 2 2 1 6 1 5, 4 (1) : 1

2 3 1 4 2 2 3 2 1 3, 1 (1) : 1

2 2 1 3 2 2 3 6 1 5, 4 (1) : 1

2 3 1 3 2 2 3 2 1 1, 5 (1) : 1

2 2 1 4 2 2 1 6 1 5, 4 (1) : 1

2 2 1 3 2 2 3 6 1 1, 5 (1) : 1

2 2 1 3 2 1 3 2 1 3, 4 (1) : 1

2 2 3 3 2 2 3 6 2 1, 4 (1) : 1

2 2 1 3 2 2 3 6 1 3, 4 (1) : 1

2 3 1 2 6 2 3 2 1 3, 6 (1) : 1

2 3 1 2 6 2 3 2 1 5, 6 (1) : 1

2 3 1 2 6 2 3 2 1 1, 6 (1) : 1

2 2 3 3 2 1 3 4 1 3, 4 (1) : 1

2 3 1 4 2 1 3 2 1 3, 3 (1) : 1

2 3 1 3 2 2 3 4 1 5, 5 (1) : 1

2 2 1 3 2 2 3 4 1 3, 4 (1) : 1

2 2 1 3 2 1 3 4 1 3, 4 (1) : 1

2 3 1 3 2 1 3 4 1 5, 5 (1) : 1

2 3 1 4 2 2 3 4 1 3, 1 (1) : 1

2 3 1 4 2 2 3 4 1 5, 5 (1) : 1

2 2 1 4 2 2 3 4 1 3, 4 (1) : 1

2 2 1 4 2 2 3 4 1 5, 4 (1) : 1

2 2 3 3 2 2 3 4 1 3, 3 (1) : 1

2 3 1 4 2 2 1 4 1 3, 3 (1) : 1

2 3 1 3 2 1 3 4 1 3, 5 (1) : 1

2 3 1 4 2 1 3 4 1 3, 2 (1) : 1

2 2 1 4 2 1 3 4 1 3, 3 (1) : 1

2 3 1 3 2 2 3 4 1 3, 2 (1) : 1

2 2 1 4 2 1 3 4 1 5, 4 (1) : 1

2 2 3 4 2 1 3 4 1 3, 4 (1) : 1

2 2 3 4 2 2 3 4 1 5, 4 (1) : 1

2 2 3 4 2 2 3 4 1 3, 3 (1) : 1