Creating a Difficulty Metric for A Sudoku Variation

Emily Alfs
Mathematics and Computer Science
Doane College
Crete, Nebraska 68333
emily.alfs@doane.edu

Abstract

Frame Sudoku is very similar to traditional Sudoku. The game is set up with 9X9 grid and nine 3X3 sub-grids. The goal of the game is the same as traditional Sudoku: place the values 1-9 exactly once in each row, column, and 3X3 sub-grid. Frame Sudoku differs from traditional Sudoku in how it starts. In Frame Sudoku, we are given only "frame clues" and no internal clues.

As with Sudoku, there are many ways to judge the difficulty of a game. During the fall semester, we created a computer program that would assess the difficulty of any given Frame Sudoku game based on the number of times any techniques were used in the solving process. We designed and implemented these techniques then weighted those techniques based on their individual difficulties. Once we created a system that would rate games, we created a machine-learning program to learn our rating system.

1 Introduction

Sudoku as we see it in newspapers and magazines is a very approachable game. You are given a nine by nine board that is separated into nine three by three sub-grids. The game starts with a certain number of cells filled in with values ranging from one through nine. The goal of the game is to get the values one through nine in each row, column, and three by three block exactly once. When we see Sudoku in the newspaper or in magazines, we often see it accompanied with some sort of difficulty rating. These can range from easy to brainy and typically, ranking techniques vary from source to source.

The goal of this research was to create a machine-learning program to learn a particular rating system we create for a variation of Sudoku, Frame Sudoku. Our rating system was set up as follows: we analyzed solving techniques for Frame Sudoku, assigned those techniques a difficulty rating, then based on how many times each technique was used we assigned a difficulty level to the puzzle. Once we had a large enough data set, we created a machine-learning program to learn our system and tested it on other data points.

2 How to Play

Frame Sudoku is similar to traditional Sudoku in the form of the game and the goal. However, the starting givens are different. There are no cells filled in initially and the only clues are on the outside of the board, the frame. These clues tell the player the sum of the three closest cells. An example of this game can be seen in Figure 1.

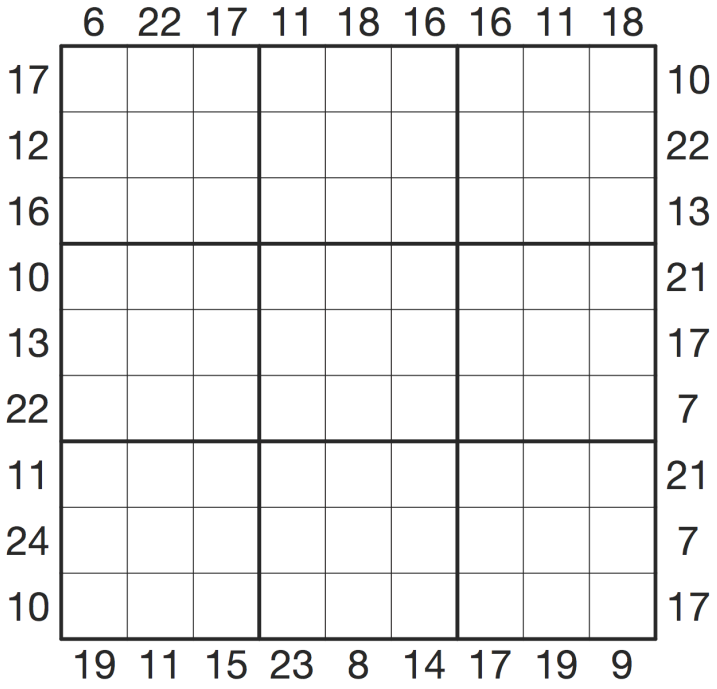

Figure 1: An example of a typical Frame Sudoku game.

As mentioned previously, each frame clue tells us the sum of the three closest cells. To help illustrate this, consider the top left block of the game in Figure 1. The 6 tells us that the values 1, 2, and 3 must go in the three cell column below the 6. However, we do not know in what order to place them. We know that these values must be 1, 2, and 3 as Sudoku rules allow us to use the values 1 through 9 exactly once in each row, column, and block. This can be better visualized in Figure 2.

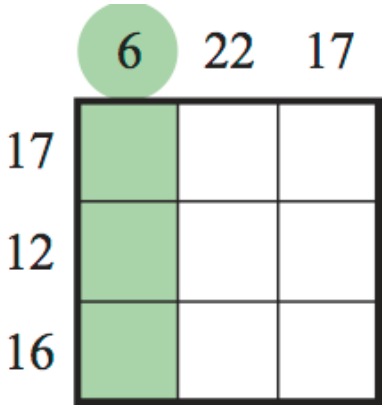

Figure 2: We know 1, 2, and 3 must go in the green cells, as those are the only values that add up to 6 with Sudoku constraints.

To figure out what order the 1, 2, and 3 must be placed in we must use the intersecting row clues. As one might be able to tell, the strategy behind this variation relies heavily on partitions. A partition is a way of writing an integer *n* as a sum of positive integers where the order of the summands is not significant, possibly subject to one or more additional constraints. [2]

The partitions of the three row clues, 17, 12, and 16, will help us to place the 1, 2, and 3 in column one. However, these will be more difficult to solve for as they have many more partitions. Consider the frame clue 12: this can be filled in using, {1, 2, 9}, {1, 3, 8}, {1, 4, 7}, {1, 5, 6}, {2, 3, 7}, {2, 4, 6}, or {3, 4, 5}. Thus, the game must be played more strategically.

3 Previous Research

During the summer, Susanna Lange and I analyzed this particular version of Sudoku to better understand it. This work was partially supported by National Science Foundation grant DMS-1262342, which funds a Research Experiences for Undergraduates program at Grand Valley State University. One of the many products of this research was a program that generated Frame Sudoku games with unique solutions, meaning they can only be filled in one way.

I continued with this topic at Doane College for a senior research project. The goal of this research was to create a difficulty metric for Frame Sudoku. We found Djape's book that had approximately 50 rated Frame Sudoku games. [1] Throughout the research, we

attempted to reflect Djape's rating system by creating our own. Our system was developed by defining solving techniques and programming them so a computer could solve games as a person would using these techniques. Difficulty was rated by the number of times each technique was used. Based on the level of difficulty, these techniques were given different weights. These weights were added together to give us our difficulty rating, which ranges from 10 to 3,400. Unfortunately, we were not able to replicate Djape's rating system. However, we were able to create unique games and assign them difficulties to train and test our machine learning system. From these previous research experiences, we were able to create and rate as many games as we would like to run through our neural network.

4 Neural Network

An artificial neuron network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output. [5]

The key components to a neural network are edges and nodes. Each edge, which is a connection between the nodes, has a weight. The assigned weight of that edge will multiply the value that is traveling down the edge. The nodes can have different types and in our case, we had two types: multipliers and adders. The nodes accumulate the weighted values that are coming from the input edges. The node either adds all of the weighted input values or multiplies them depending on the node type.

Neural networks consist of three layers: the input layer, one or more hidden layers, and the output layer. The input layer is just that, the initial input values. In our case, the input values were all of the frame clues for a single game. The hidden layer is where the weights and accumulations happen which was mentioned previously. To see the specific setup, please refer to Figure 4 at the end of this paper, which shows half of the neural network. Our output layer is a single node that represents our difficulty.

In order to find the weights and node types, we had to "train" our neural network. Our training process was through a genetic algorithm.

5 Genetic Algorithm

We modeled natural evolution in a genetic algorithm by using operators such as crossover, mutation, and selection to get the best individual. Crossover behaves like a mating process in that it takes two individuals, switches some of their DNA, and produces two new offspring. Mutation takes a single individual and randomly changes its DNA based on a probability of mutation. Selection takes two random individuals, evaluates their fitnesses, and selects the individual with the better fitness [3]. This process

trained our population of neural networks to ultimately give us the fittest individual, meaning the most accurate neural network.

The genetic algorithm we created was based off of the Doane Evolutionary Algorithm (DEA) [4]. In our case, the individual, a single neural network, is represented as an array of 107 doubles. This allowed us to use methods that were built into the DEA for crossover, selection, and mutation. Our neural networks had a static structure so they always had the same number of nodes and edges, half of which can be seen in Figure 4 at the end of the paper.

5.1 Crossover

Crossover happens at a rate of 60%. So we go through the entire population and generate a random number between 0 and 1. If that number is below .6 then crossover happens. When an individual is selected for crossover then another random individual is selected and they crossover a random value in their respective arrays.

5.2 Mutation

Our mutation operator is rather straightforward. The mutation will happen if a random number between zero and one is less than our chosen μ, which we have set to .25. Our mutation is a single point mutation so only one value in our array will change when mutation happens.

5.3 Selection

Within our program, we use elitist tournament selection. This ensures that our best individual from the population survives into the next generation. Without elitist tournament selection, this is not guaranteed. In regular selection two individuals from the current population are selected and have their fitness's compared. Whichever individual has the higher fitness goes to the next generation. However, both individuals remain in the current population and are subject to selection again. By using elitist tournament selection, our next generation will always contain the best member of the previous population.

5.4 Fitness

Fitness of a single neural net is measured by putting 500 Frame Sudoku games through the neural net. These games were produced and rated from previous research and have an expected difficulty level already assigned to them. We then add together the differences between the expected difficulty level and output layer value of the game.

6 Results

Half of our fittest neural network can be seen in Figure 4 at the end of this paper. This individual was reached using 2000 generations each with a population size of 100,000 neural networks. Figure 5 shows a partial table of results. These results are based on games that the neural network was not trained on.

Overall, the average difference between the expected value of the game and the evaluated value was 453. The standard deviation was 578 and the variance was 335,124. Some ways that we could improve the results would be to run the genetic algorithm with more generations and individuals or to change the structure of the neural network. As mentioned previously, our neural network was a static structure so it never changed. The structure of the neural network could potentially evolve through the genetic algorithm as more advancement occurs throughout this project.

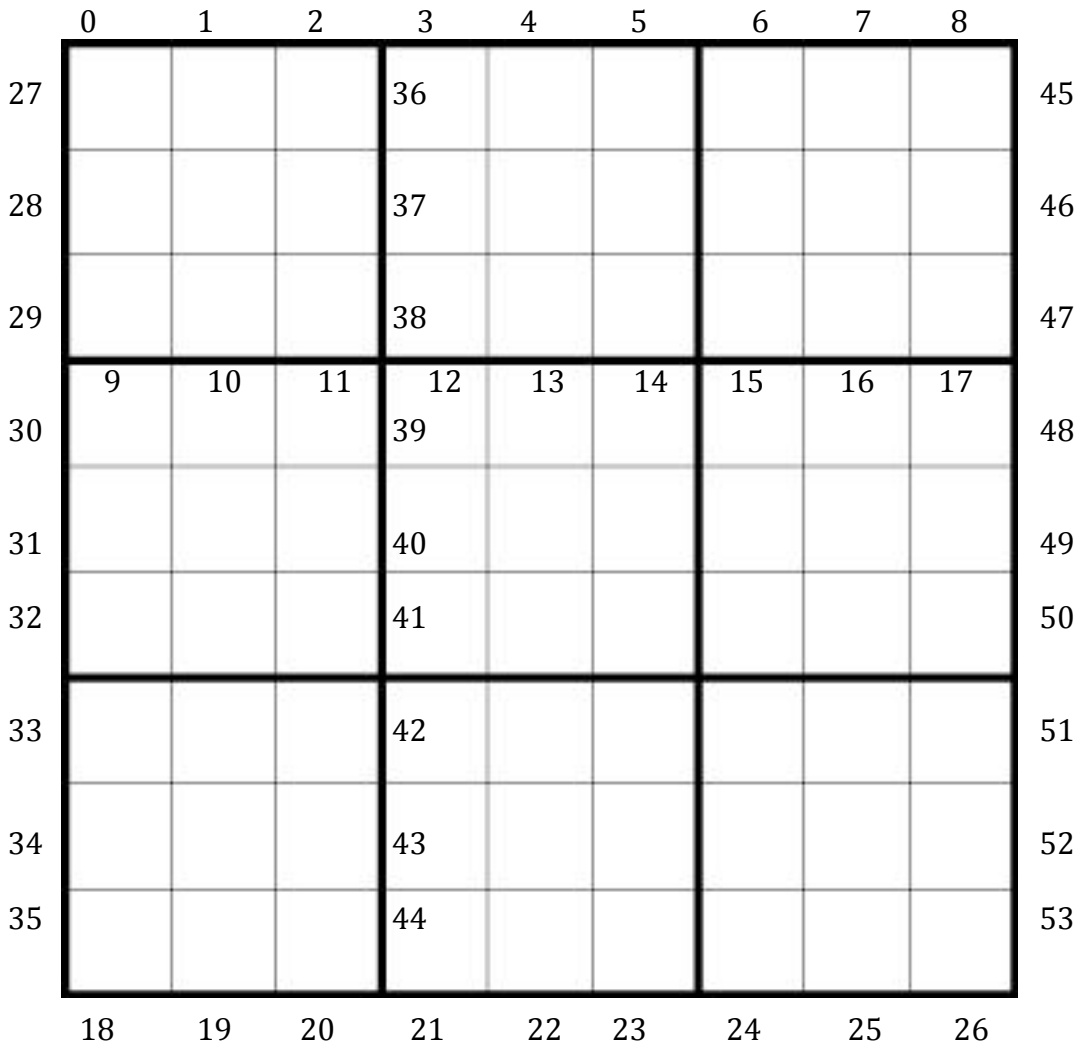

Figure 3: This diagram simply shows how the frame clues map to the input layer of the neural network.

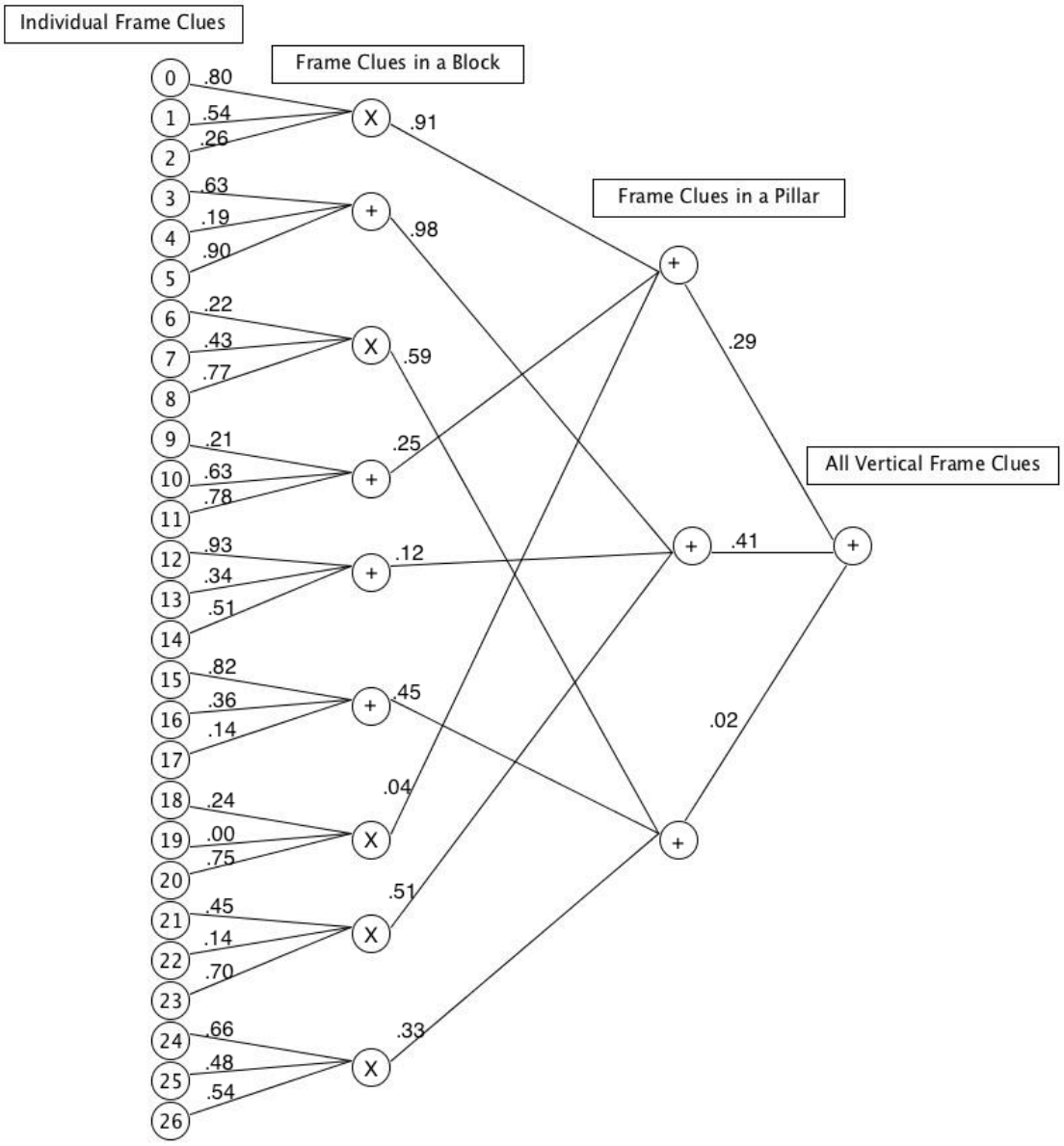

Figure 4: This shows half of the neural network structure and half of the assigned values. The other half is symmetric in shape however the values are different. The two halves would be joined with two edges to a single node, which is our output value.

Expected Difficulty	Evaluated Difficulty	Difference
33	16.08077955	16.91922045
19	13.59114764	5.408852356
950	15.10677381	934.8932262
16	15.1720173	0.827982701
405	15.10711983	389.8928802
470	14.12369419	455.8763058
107	15.95362926	91.04637074
23	15.82859032	7.171409676
33	15.23634969	17.76365031
972	16.53997834	955.4600217
27	15.14759752	11.85240248
1499	14.80204318	1484.197957
1700	13.90621677	1686.093783
64	14.5081555	49.4918445
14	15.71562718	1.715627182
27	14.68313603	12.31686397
18	14.7443567	3.2556433
16	15.74902504	0.250974962
171	16.429595	154.570405
21	15.23400112	5.765998878
21	15.85727046	5.142729535
20	14.64316211	5.356837888
2300	15.09437453	2284.905625
29	15.88480141	13.11519859
25	15.78036096	9.219639042
31	15.27467276	15.72532724
13	15.85984992	2.859849923
486	16.2923655	469.7076345
513	14.50608174	498.4939183
189	14.04382103	174.956179
44	15.36066631	28.63933369

Figure 5: Partial results depicting the expected value, which was assigned by a program using solving techniques, then the value evaluated by the neural network, followed by the difference.

References

[1] Djape, Frame Sudoku: A Hybrid Between Killer Sudoku and Outside Sudoku, CreateSpace Publishing, 2014
[2] Hardy, Wright, Encyclopedia of Mathematics, 2003
[3] Hiu Man Wong. Genetic Algorithms. In *SURPRISE 96 Journal*, 1996
[4] Mark M. Meysenburg. The DEA: A Framework for Exploring Evolutionary Computation. In *MICS 2004: Proceedings of the Midwest Instruction and Computing Symposium*, April 2004.
[5] Techopedia, Artificial Neural Network