IT3708 - Project 2

Evolving Spiking-Neuron Parameters

Written by: Thomas Almenningen, Halvor G. Bjørnstad

Introduction

The code is written in C++. In order to generate the plots you need to have matlab installed. We have added the plots and some of the figures in the .zip file for those who wish to view larger versions.

1. System Overview

1.1 Class Diagram

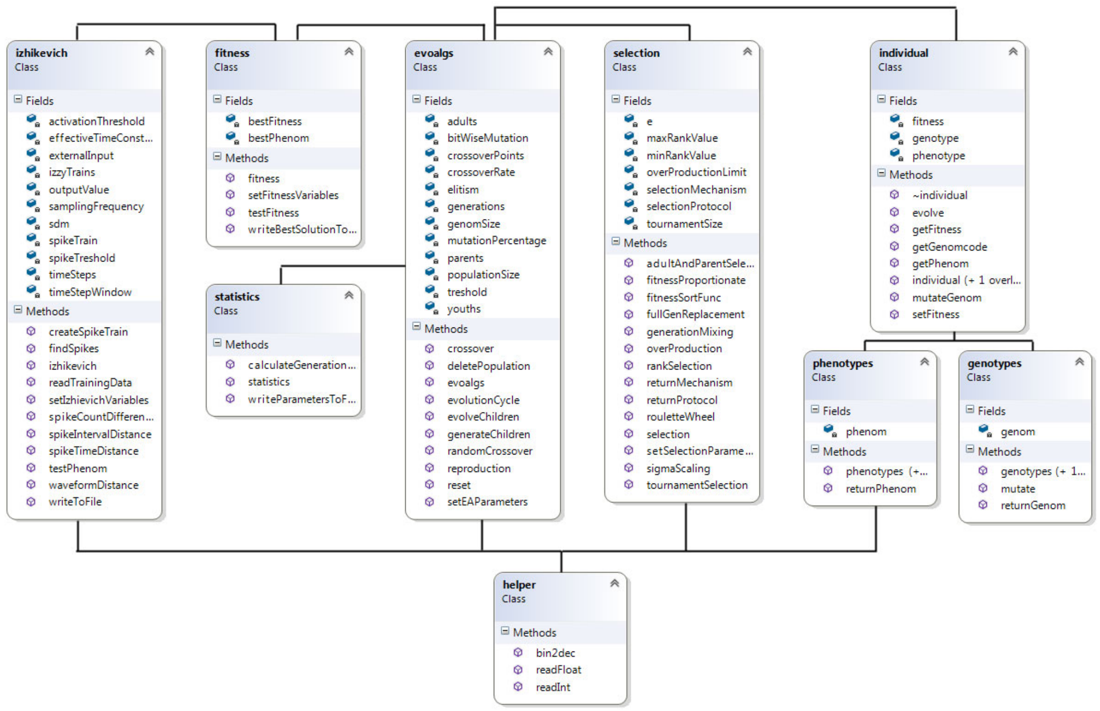


Figure 1: Class Diagram Generated in Microsoft Visual Studio 2012 Professional

The system consists of the evolutionary algorithm core and an additional class Izhikevich which contains most of the problem specific code. The Izhikevich class is used as an extension of the fitness class to generate the spike trains and compute the similarity to the target based on the selected metric before returning it to our fitness class. We also extended the fitness class to keep the best found solution at all times.

We changed some parts of the algorithm for it to function better with our problem specific genotype structure. The random crossover function now selects a random crossover point which keeps our genes intact instead of choosing an arbitrary crossover point.

Our mutation function was rewritten from using a bitwise mutation percentage to mutating a random gene's decimal value.

```
decimal = gHelper.bin2dec(temp);
maxMutation = ceil(MAX_SEGMENT_VALUE*mutationPercentage + 0.5f);
mutation = (rand()%maxMutation)+1;
decimal += (rand()%2) ? mutation : -mutation;
```

We first calculate the gene's decimal value before we find the maximum value that can be added or subtracted to that value. maxMutation depends on the mutationPercentage and the maximum number representable by the number of bits in the gene. We then generate a random number between 1 and maxMutation and then add or subtract that number to our current gene value before updating the genotype.

We implemented a mechanism to combat stagnation of the maximum fitness. The solution we implemented records the generation count each time there is made an improvement in the maximum fitness. For each Kth generation with no improvement we increase the multiplication factor added to the mutation percentage by 1 (Initially 1). Lets say that K=15. If there have been 30 generations since the last time there was made an improvement to the maximum fitness we multiply the mutation percentage by a factor of 3 (1 + floor(30/15 + 0.5f)).

1.2 Genotype Representation

Starting off we chose a genotype representation with a total of 33 bits. The genotype was separated into five genes which each corresponds to a parameter in the phenotype. When we tried running the program we could find okey solutions, but not great. The solution turned out to be to increase the number of bits in order to increase the number of different values you could represent within the range for each parameter, thus making us able to find more accurate solutions.

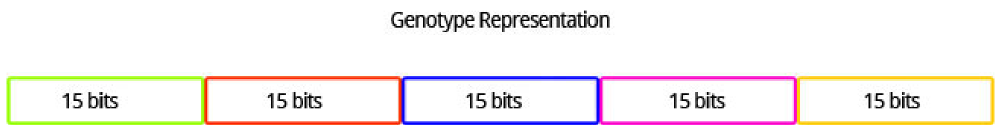


Figure 3: Genotype Representation, Fixed length linear object, Data Oriented (Classical Bit Vector)

By using 15 bits to represent each gene it allows us to represent 2^{15}-1 (32767) different values for each parameter.

1.3 Fitness Function

The fitness function in our case is fairly simple. Depending on the chosen similarity metric you get a value back. The final fitness is then calculated as following:

$$fitness = \frac{1}{1 + SDM\ Result}$$

Which constrains our fitness values between [0, 1].

2 Test Cases

The whole idea by using this tool is to find as highly optimized results as possible, but in order to do this we found that you had to use fairly high population sizes and generation count (Which would mean a lot of waiting!). After some discussion we ended up using the following parameters for all our runs, which seemed to give a good enough trade off between the time spent waiting for the algorithm to finish and the results we got out.

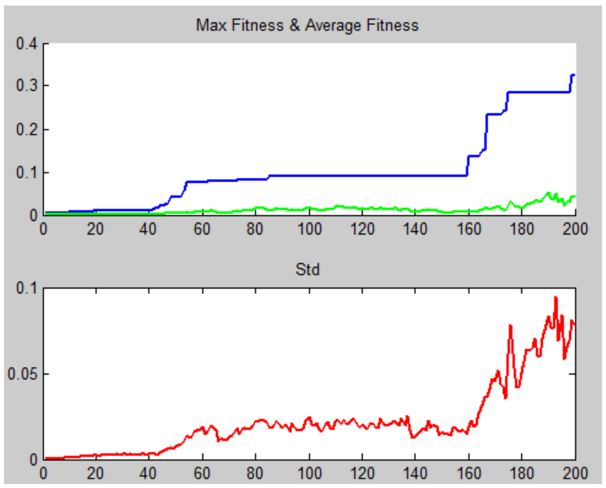
Parameters Used on all test cases:

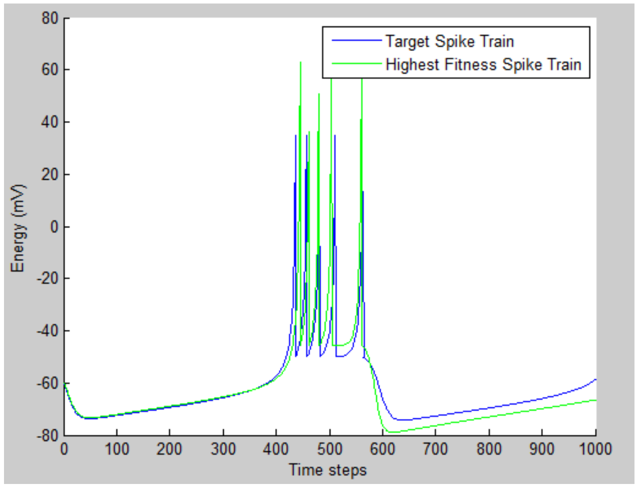
- Population Size: 100
- Generation Count: 200
- Elitism: 1
- Selection Mechanism: Sigma-Scaling
- Selection Protocol: Full Generational Replacement

NB! When we ran these tests we still calculated the fitness value by taking 1/(Similarity Metric Result), which we later changed to 1/(1+Similarity Metric Result) in order to bound or max fitness between [0, 1]. The fitness values you see on some of the plots can therefore be greater than 1. The implications of the change were so small that we choose not to run the experiments all over again (We hope that this is ok).

2.1 Spike Time Distance Metric

2.1.1 Test Data 1

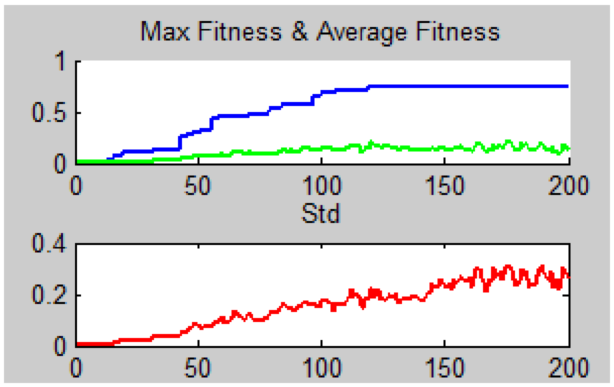


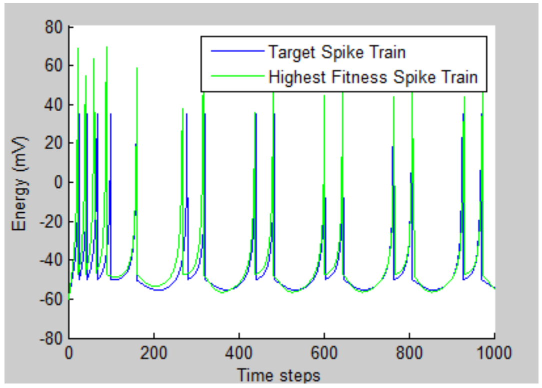


Parameters: Crossover chance: 0.2, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.1
Izhikevich Parameters: a = 0.028554 b = 0.156049 c = -45.3874 d = 3.38328 k = 0.040032

Some of the curve heights are slightly off, but apart from that we reached a good solution if we mainly focus on the spike positions.

2.1.2 Test Data 2

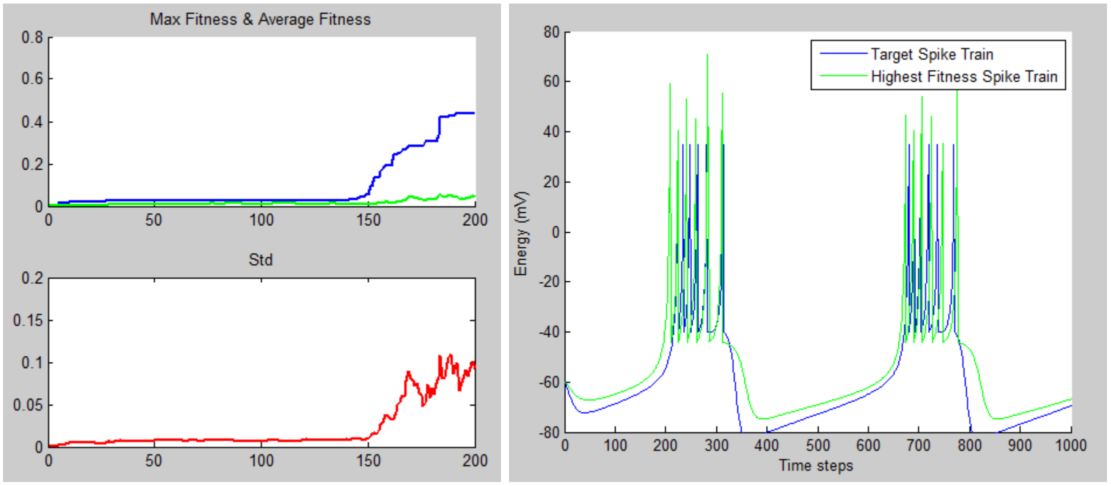




Parameters: Crossover chance: 0.2, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.1
Izhikevich Parameters: a = 0.0234343 b = 0.3 c = -47.3711 d = 5.80699 k = 0.0470717

Reusing the parameters from test 1.1 provided a great result without any need to tweak. Again, curve heights are slightly off, but distances and positions are pretty much spot on.

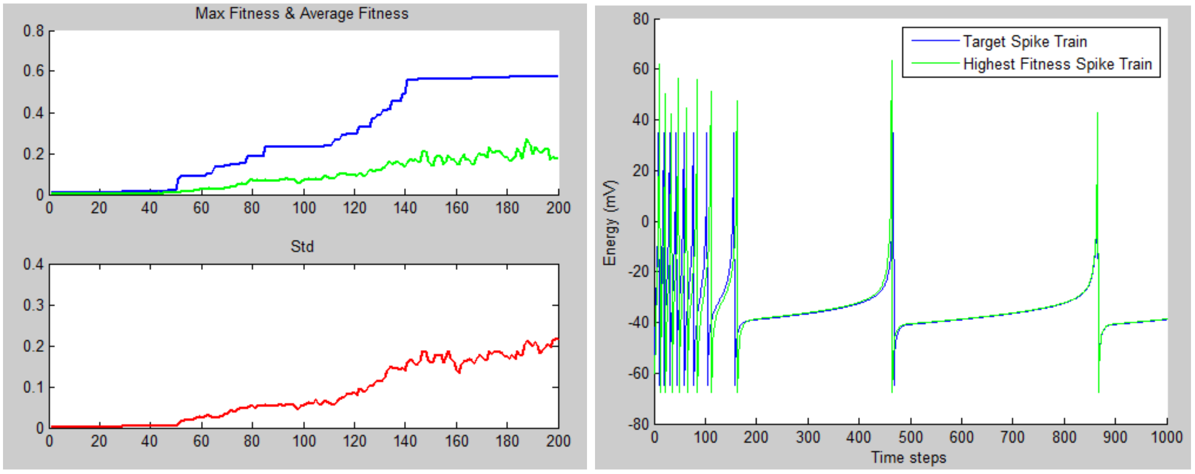
2.1.3 Test Data 3



Parameters: Crossover chance: 0.2, Mutation Chance: 0.7 and Max. Mutation Percentage: 0.15.
Izhikevich Parameters: a = 0.0445751 b = 0.139481 c = -44.037 d = 2.98175 k = 0.0408176

Initially we found poor local maximums of 0.1 fitness, so we upped the mutation rate and the mutation in order to find more optimized solutions. The spike positions fit well enough, but there is still a lot of room for improvement.

2.1.4 Test Data 4

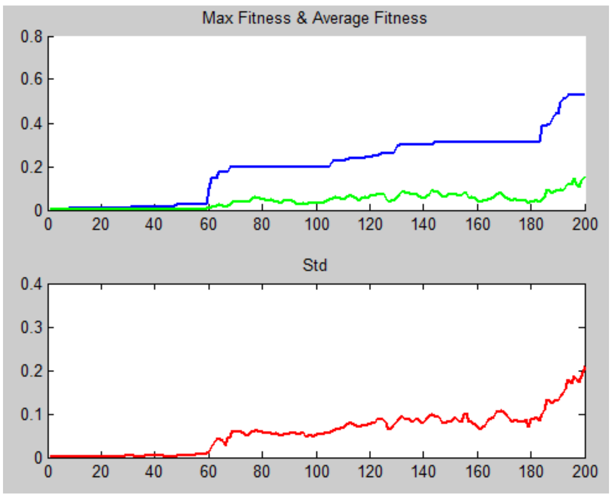


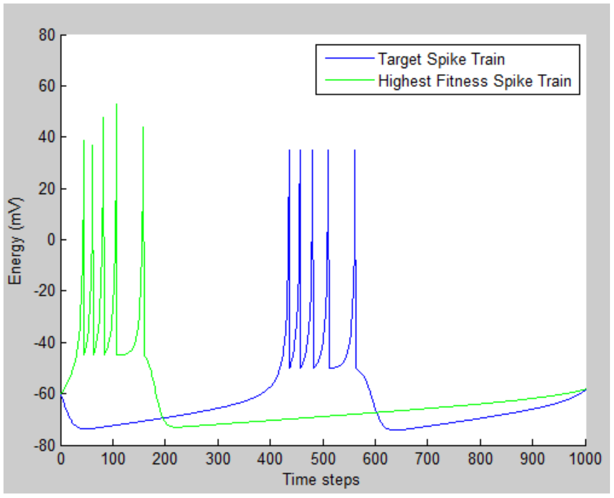
Parameters: Crossover chance: 0.2, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.1
Izhikevich Parameters: a = 0.00310132 b = 0.14488 c = -67.5851 d = 10 k = 0.0804575

Reusing the parameters from the first two tests immediately produced a very nice result in our opinion, again the heights of the curves are a bit too high, but this seems like the trend when using this metric.

2.2 Spike Interval Distance Metric

2.2.1 Test Data 1

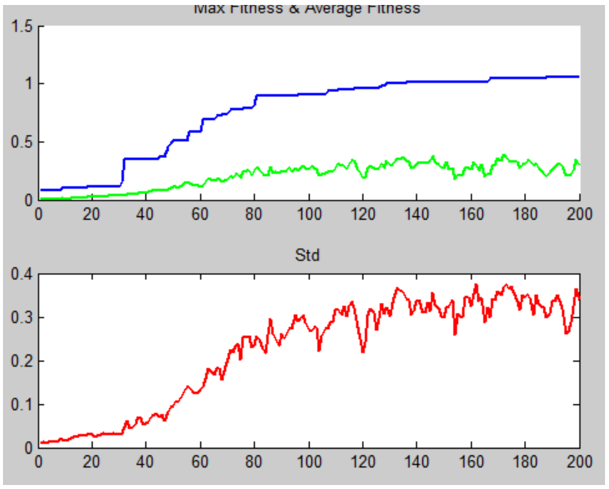


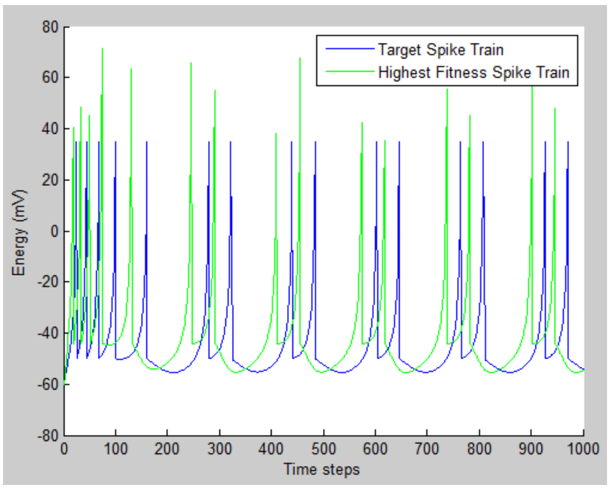


Parameters: Crossover chance: 0.2, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.1
Izhikevich Parameters: a = 0.0119925 b = 0.0454457 c = -44.9251 d = 3.06121 k = 0.042721

Although it might seem wrong at first sight we must take into account that the spike interval distance metric completely ignores spike positions. The relative curves and positions between the spikes are almost equal except for slight height differences.

2.2.2 Test Data 2

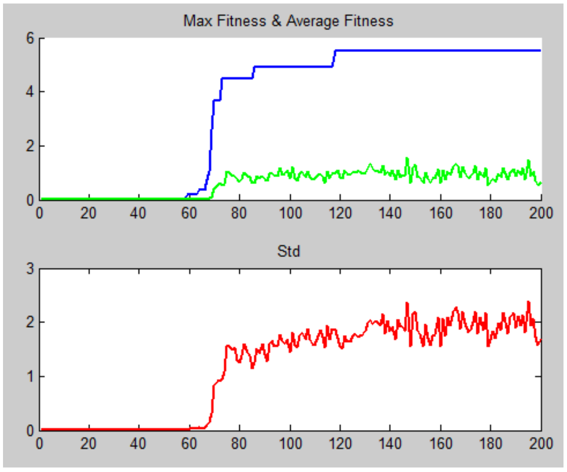


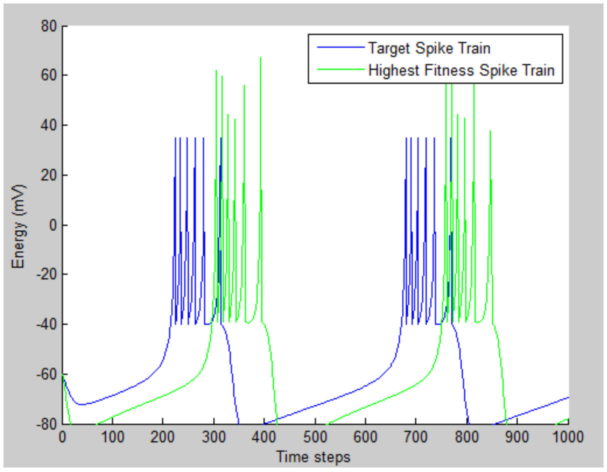


Parameters: Crossover chance: 0.6, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.12
Izhikevich Parameters: a = 0.034524 b = 0.0689789 c = -44.0156 d = 7.62765 k = 0.0497909

A very good result in our opinion, again the spike heights are a bit off, but the curves and the distances are pretty much equal. We struggled a bit finding a set of parameters that would produce a decent result, mainly having problems with too flat increases in fitness.

2.2.3 Test Data 3

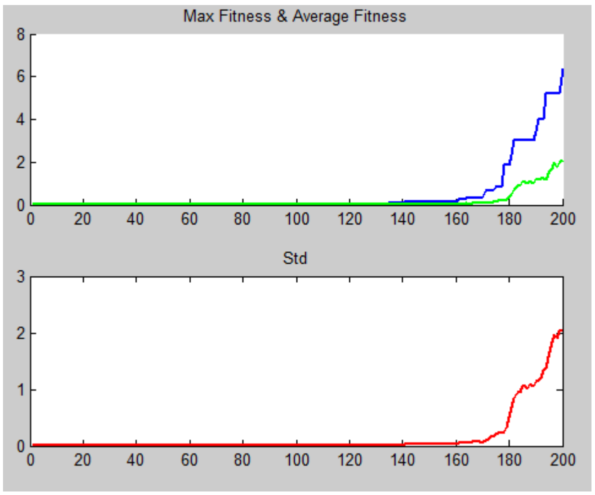


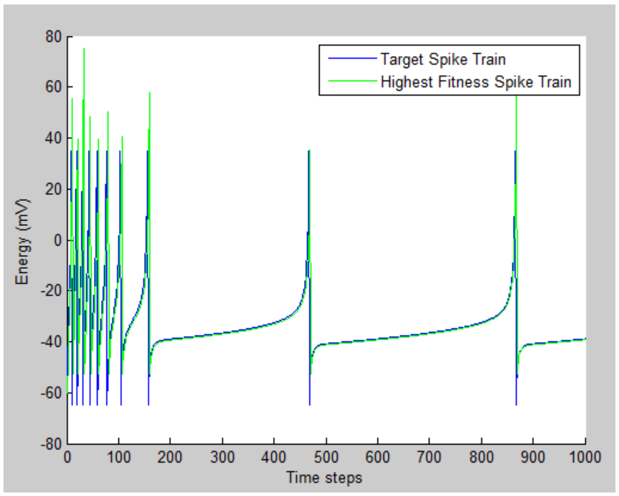


Parameters: Crossover chance: 0.3, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.12
Izhikevich Parameters: a = 0.0828179 b = 0.271307 c = -39.0518 d = 6.25536 k = 0.0377056

Again some struggling was required, but we found a more than adequate result. Heights are again a bit off, but the spike distances seems very reasonable.

2.2.4 Test Data 4



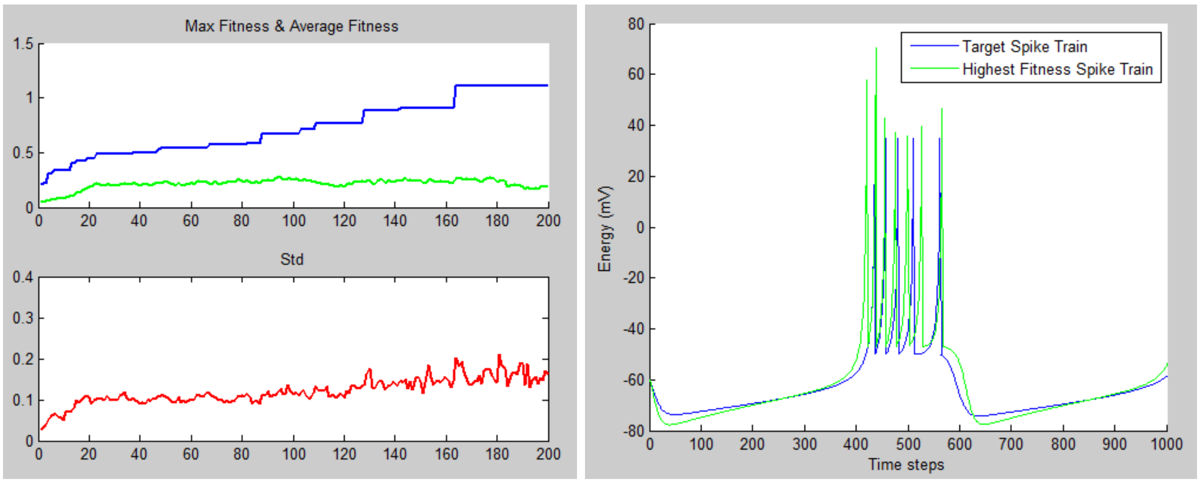


Parameters: Crossover chance: 0.3, Mutation Chance: 0.6 and Max. Mutation Percentage: 0.15
Izhikevich Parameters: a = 0.00304666 b = 0.219391 c = -52.6218 d = 9.85316 k = 0.0790375

The fitness plot is quite strange, but we have no complaints regarding the best spike train we produced. This particular problem suits this metric fairly well due to the fact that there are very few ways you can represent a similar graph in the same time window. We speculate that a higher mutation percentage would have provided us with a prettier fitness graph.

2.3 Waveform Distance Metric

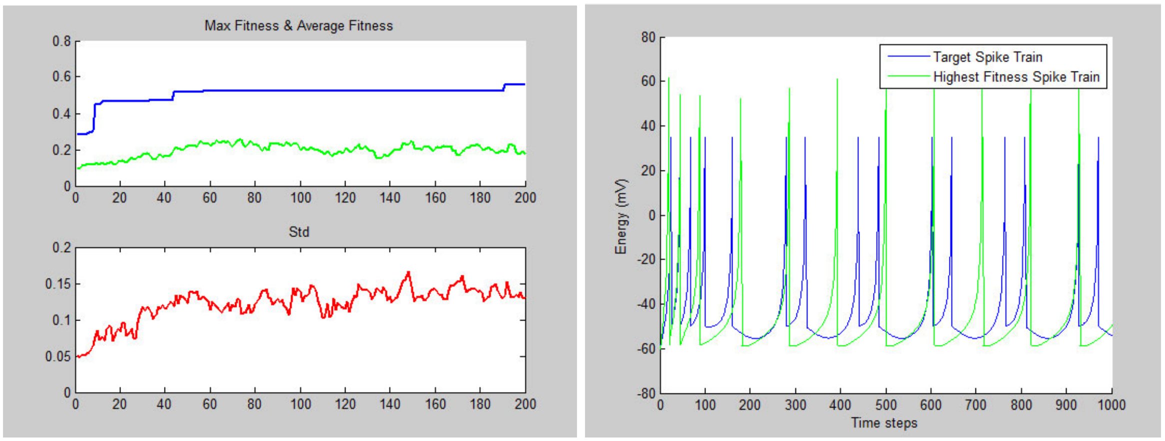
2.3.1 Test Data 1



Parameters: Crossover chance: 0.25, Mutation Chance: 0.65 and Max Mutation Percentage: 0.15 Sampling Distance 5
Izhikevich Parameters: a = 0.0564421 b = 0.179936 c = -46.8416 d = 2.1817 k = 0.0390954

Great steep fitness curve, a very good resulting spike train. We had to increase mutation and crossover variables. The particular spike train seemed to be handled very well by the waveform metric compared to the others.

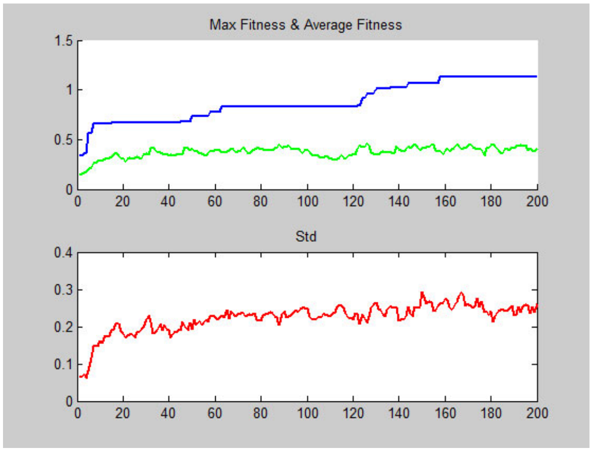
2.3.2 Test Data 2

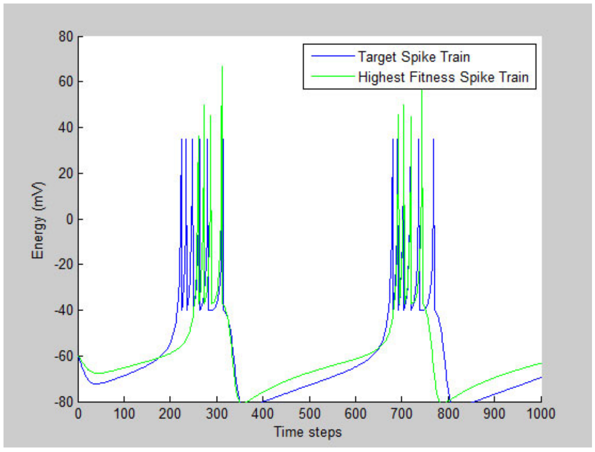


Parameters: Crossover chance: 0.25 Mutation Chance: 0.5 and Max. Mutation Percentage: 0.15, Sampling Distance 25
Izhikevich Parameters: a = 0.041332 b = 0.01 c = -58.4021 d = 10 k = 0.049368

For this particular target spike train we had to increase the sampling density to avoid getting a result with way too many fluctuations. Some of the plots we got using a lower sampling distance literally turned the whole graph green. But when using a higher sampling distance we were able to find a ok result, but far from good. By far the hardest problem for the waveform metric.

2.3.3 Test Data 3

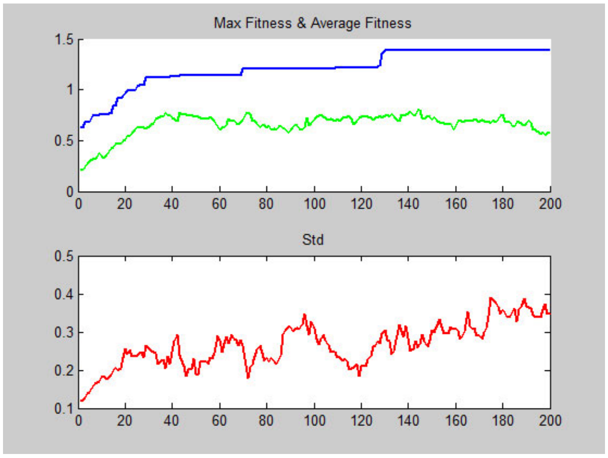


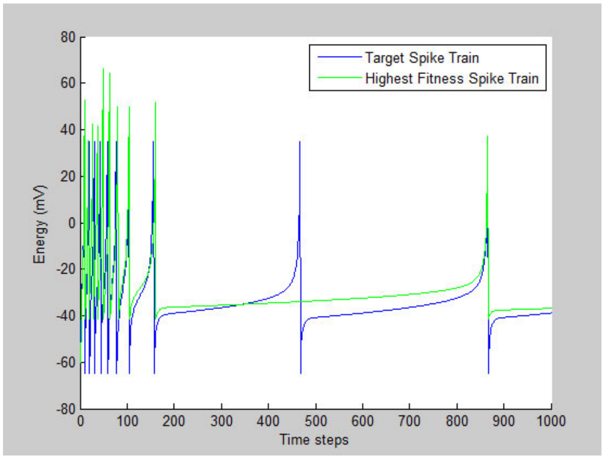


Parameters: Crossover chance: 0.3 Mutation Chance: 0.75 and Max. Mutation Percentage: 0.1, Sampling Distance 5
Izhikevich Parameters: a = 0.132867 b = 0.0756609 c = -37.0009 d= 9.35223 k = 0.0406665

A totally reasonable result, but again. Not great. The generated spike train follows the contour good enough, but lack some of the fine details. We tried to play around with the sampling distance, but we didn't get any significantly better result when setting it to 1.

2.3.4 Test Data 4





Parameters: Crossover chance: 0.2 Mutation Chance: 0.6 and Max Mutation Percentage: 0.1 Sampling Distance 5
Izhikevich Parameters: a = 0.00132188 b = 0.0415427 c = -41.2094 d = 9.7731 k = 0.0895215

Another spike train that the wave distance metric seemed to struggle with. The result is again ok, but very far from a good eyeball result. By using a low sampling distance we were able to capture the fine detail at the start and one of the following spikes.

2.4 Conclusion

The different metrics all have their individual strengths and weaknesses. The Spike Time Distance Metrics gave us results where the spike positions correlates very well, and normally gave us the best eyeball result. However, it struggles more with parts of the graph which are not considered spikes. The height of the spikes were also off on most of our graphs.

The Spike Interval Distance Metric works well on some problems, and worse on others. The graphs were only a small portion of the graph consists of spikes it could generate the spikes where the distances between the spikes correlated well with the target, but with a huge time offset. For other problems where the spikes occurred over the bigger parts of the graph it gave us better results, since there was a smaller room for time offset.

The Waveform Distance Metric gave us some reasonable results, but due to the fact that it used a reduced number of sampling points, it is easy to miss some of the fine details. And it often generated some totally absurd results when using a very low sampling distance. But generally we were able to find fairly reasonable results.

We were not able to find any perfect solutions, but we suspect that we should have implemented a measure to reduce the amount of mutation when we started getting closer to a very good result (Feks. a fitness value above 0.2), but again. This has its perils. By doing this you might end up converging more often on a local maxima. It would also have been preferable to implement multiple similarity measures to get a more complete fitness value.

3. Classification of Genotype-Phenotype Mapping

The genotype to phenotype mapping used in this project can be classified as a generative encoding scheme. This means that our genes encode parameters for a developmental scheme. More specifically this means that our entire genotype is separated into multiple genes which each encode a separate parameter. Each gene is then run through a genotype to phenotype mapping function which converts each gene into a decimal value that represents a parameter in our phenotype. These parameters are then used in a developmental process function to generate a spike train.

4. Practical Implications

Evolutionary algorithms are well suited for optimization problems such as this one. And can therefore be used to evolve highly optimized parameters. The izhikevich model allows us to reproduce firing behavior very similar to what occurs in the biological brain. Used together these tools allow computational neuroscientists to reproduce many different firing behaviors that can occur in biological spiking neurons. This information can then be used to model and study the firing patterns of single neurons, allowing a computational neuroscientist to do countless simulations on the same data. This model can then be used by students and others who are interested in learning more about firing patterns of single neurons but doesn’t have access to the necessary equipment.

5. Other Problem Domains

Due to the fact of the computational efficiency of the izhikevich model you could use a more general version of this tool you can compute the optimal parameters for multiple neurons and build large-scale networks of spiking neurons. Which in turn might help us unravel some of mysteries of our brain or at least give us a better understanding of the brains dynamics.

One example of another problem domain might include:

By using training data (Membrane Potentials) collected from patients with a brain deficiencies or diseases (like parkinson), you can study how different brain disorders affect the firing patterns of neurons. By building networks with “sick” neurons you will also be able to study the implications the disease have on large-scale networks of neurons.

One of the prerequisites to test out one such system would be to have a target model built using training data from a healthy individual. Preferable the same the same patient, one alternative would be to build different models as the disease progresses and use these as the target models.