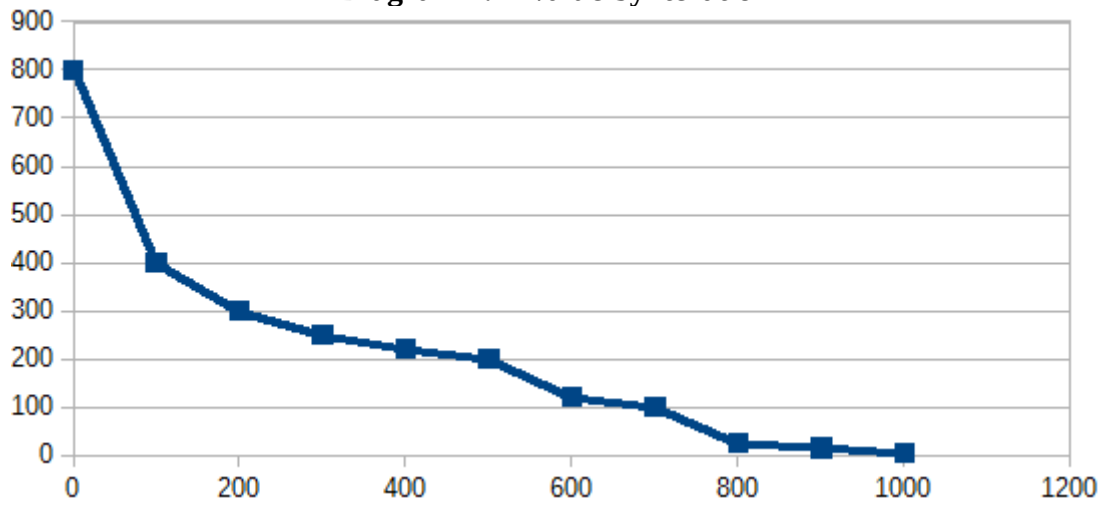
Dzianis Palchuk. ii-10.CIIT practice. Lab 6
Hinton and Nowlan's Simulation of Life Time learning

Diagram 1. N value by iteration



Source code:

```
import random
from math import sin
def get_average_value(generation):
    av_list = []
    for i in generation:
        sum = fitness_function(i)
        av_list.append(sum)
    average = reduce(lambda x, y: x + y, av_list) / len(av_list)
    min_value = min(av_list)
    return average, min_value
def fitness_function(i):
    return getResX(i)*sin(abs(getResX(i)))
def getResX(i):
    sum = 0
    for count in range(0,10):
        if i[count] == 1:
            sum += 2**count
    if i[0] == 1:
        sum *=-1
    return float(sum)/1023*5
def main():
    generation = []
    for i in range(0, 20):
        hromosome = []
        for j in range(0, 10):
            hromosome.append(random.getrandbits(1))
        generation.append(hromosome)
    min_value = get_average_value(generation)[1]
    print(min_value)
    count = 0
    Xs = []
    MAXs = []
    while count < 10:
        Xs.append((map(lambda x: getResX(x), generation)))
        generation.sort(key=lambda x: fitness_function(x), reverse=False)
        fathers = generation[0:10]
        new_generation = []
        for i in range(0, 10):
            mother = random.randrange(0, 10)
            father = random.randrange(0, 10)
            rand_index = random.randrange(0, 10)
```

```
            first = fathers[mother][0:rand_index]
            first.extend(fathers[father][rand_index:10])
            second = fathers[father][0:rand_index]
            second.extend(fathers[mother][rand_index:20])
            new_generation.append(first)
            new_generation.append(second)
        new_generation.extend(fathers)
        new_max = get_average_value(new_generation)
        MAXs.append(new_max)
        if new_max[0] < min_value:
            min_value = new_max[0]
            count = 0
        else:
            count += 1
        generation = new_generation
    print(Xs[0])
    print(Xs[2])
    print(Xs[len(Xs)-1])
    print(MAXs)
    print("------------------")
    print(generation[0])
if __name__ == '__main__':
    main()
```