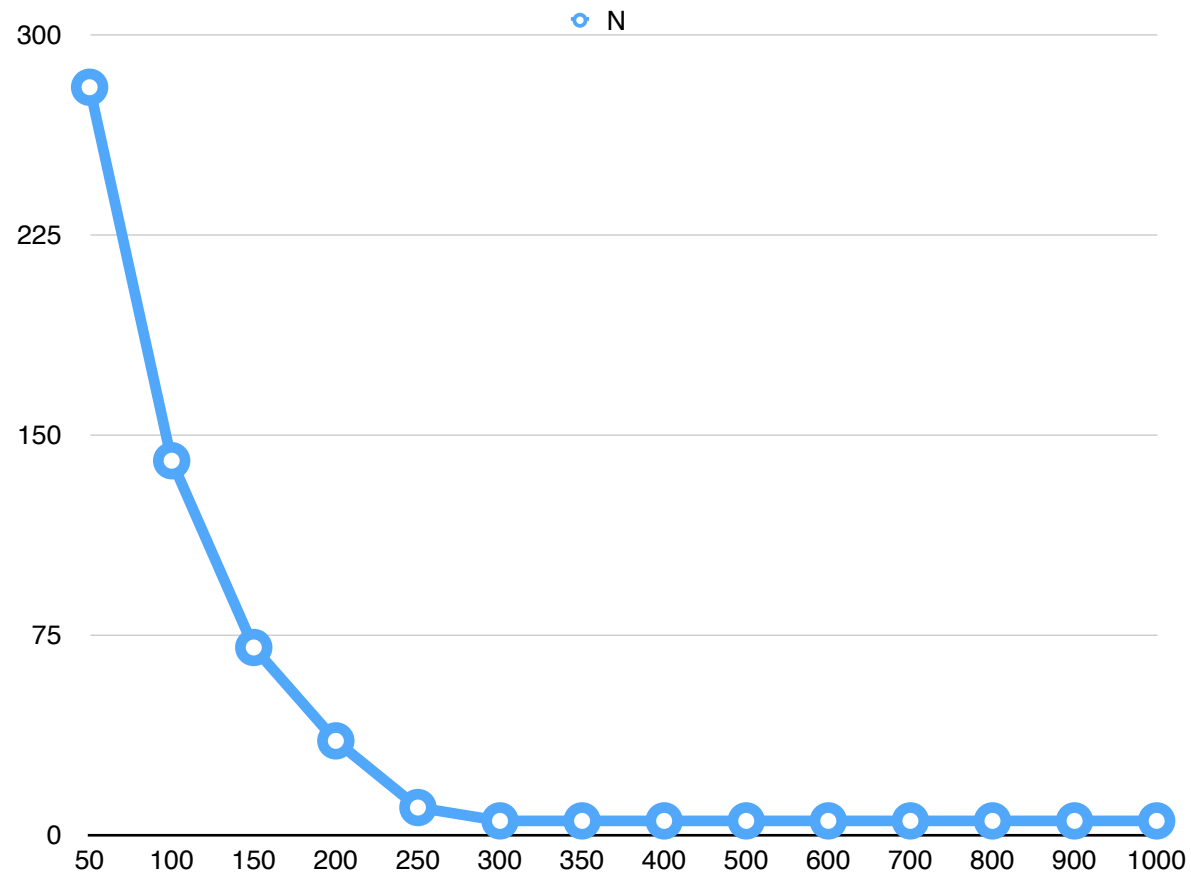
**Hinton and Nowlan's Simulation of Life Time Learning**

**Student – Ihar Yachnik(AI - 10)**

**N - y**
**Iteration - x**




Source code:

```
'use strict';


const rand = () => {
  const res = Math.random();
  return res > 0.5 ? 1 : 0;
};

const getRandomInt = (min, max) => {
  return Math.floor(Math.random() * (max - min + 1)) +
min;
}
```

```
const generatePopulation = (count) => {
  return Array.from({ length: count }, v =>
generatePerson(genesCount));
};

const generatePerson = (count) => {
  return Array.from({ length: count }, v => rand());
};

const countFitness = (array) => {
  return array.reduce((p, c) => c == 1 ? ++p : p, 0);
};

const rotatePersons = (population) => {
  return population.sort((a, b) => countFitness(b) -
countFitness(a));
}

const getStatistics = (population) => {
  maxFitness.push(population.map(e =>
countFitness(e)).shift());
  averageFitness.push(getAverage(population));
}

const getAverage = (population) => {
  const all = population.reduce((p, c) => p +=
countFitness(c), 0);
  return all / personCount;
}

// reduce less fitness persons
const trimPersons = (population) => {
  return population.slice(0, sliceCount);
};

const getParents = (population) => {
  const firstParentindex = getRandomInt(0, sliceCount -
1);
  const secondParentindex = getRandomInt(0, sliceCount -
1);

  return {
    first: population[firstParentindex],
    second: population[secondParentindex],
  };
```

```
};

const crossover = (parents) => {
  const children = [parents.first, parents.second];
  const delimiter = getRandomInt(1, genesCount);

  children.push(parents.first.slice(0,
delimiter).concat(parents.second.slice(delimiter)));
  children.push(parents.second.slice(0,
delimiter).concat(parents.first.slice(delimiter)));

  return children;
};

const main = () => {
  let population = generatePopulation(personCount); //
init

  for (let i = 0; i < iterationCount; i++) {
    population = rotatePersons(population);

    getStatistics(population);

    population = trimPersons(population);

    let parents;
    let newPopulation = [];

    for (let j = 0; j < sliceCount / 2; j++) {
      parents = getParents(population);
      const children = crossover(parents);
      newPopulation = newPopulation.concat(children);
    }

    population = newPopulation;
  }

  console.log('average', averageFitness);
  console.log('max', maxFitness);
}

main();
```