Oksana Zanko
Task 1

We have the initial population with100 binary chromosomes with 1000 genes. The fitness is the number of “1” is good, “0” is bad. We need to get gene which consists all “1”.

Algorithm:

1. Create 100 binary-chromosomes at random where each chromosome contains 1000 gens.
2. Select two chromosomes at random from the better half of the population of 100 chromosomes.
3. Create a child chromosome by a crossover. Compare two performances one with one-point-crossover and the other with uniform-crossover.
4. Randomly determine the number. If this number(A) is less than 100, we find the number(B) with number(A) and if number(A) is “1”, we change on “0”.
5. Pick the best chromosome (where fitness is higher).
6. Repeat steps 2-5, until we get new population.
7. If there is a chromosome with all “1”, gens in population then stop. Otherwise repeat steps 2-6

ITER.	1			10			50			90			100		
	FITNESS	PROBILITY	CHOICE	FITNESS	PROBILITY	CHOICE	FITNESS	PROBILITY	CHOICE	FITNESS	PROBILITY	CHOICE	FITNESS	PROBILITY	CHOICE
0	541	0.010813296	5	594	0.010115	5	683	0.010175	4	713	0.010111	4	713	0.010101	0
1	535	0.01069337	5	591	0.01007	5	682	0.010175	4	713	0.010111	4	713	0.010101	0
2	532	0.010633407	4	591	0.010587	3	682	0.010175	4	713	0.010111	3	713	0.010101	0
3	529	0.010573444	4	590	0.010534	3	682	0.010175	1	713	0.010111	5	713	0.010101	0
4	529	0.010573444	4	590	0.010534	4	682	0.010175	1	713	0.010111	5	713	0.010101	0
5	524	0.010473506	3	585	0.010516	5	682	0.010175	2	713	0.010111	5	713	0.010101	0

6	523	0.010453519	3	585	0.010516	2	682	0.01016	3	713	0.010111	5	713	0.010101	0
7	522	0.010433531	3	585	0.010427	2	681	0.01016	3	713	0.010111	3	713	0.010101	0
8	522	0.010433531	1	584	0.010427	1	681	0.01016	3	713	0.010111	4	713	0.010101	0
9	522	0.010433531	2	583	0.010427	0	681	0.01016	4	713	0.010111	4	713	0.010101	0
10	521	0.010413544	2	583	0.010409	0	681	0.01016	1	713	0.010111	4	713	0.010101	0
11	521	0.010413544	2	580	0.010391	0	681	0.01016	1	713	0.010111	5	713	0.010101	0
12	520	0.010393556	1	580	0.010391	2	681	0.010145	0	713	0.010111	1	713	0.010101	0
13	520	0.010393556	1	580	0.010338	2	680	0.010145	0	713	0.010111	2	713	0.010101	0
14	520	0.010393556	1	580	0.010338	3	680	0.010145	0	713	0.010111	3	713	0.010101	0
15	519	0.010373568	1	577	0.010338	3	680	0.010145	1	713	0.010111	3	713	0.010101	0
16	517	0.010333593	0	577	0.010338	1	680	0.010145	1	713	0.010111	4	713	0.010101	0
17	517	0.010333593	0	577	0.010284	2	680	0.010145	2	713	0.010111	0	713	0.010101	0
18	516	0.010313606	0	577	0.010284	3	680	0.010145	2	713	0.010111	0	713	0.010101	0
19	516	0.010313606	1	577	0.010284	0	680	0.010145	3	713	0.010111	3	713	0.010101	0
20	516	0.010313606	1	577	0.010284	0	680	0.010145	3	713	0.010111	4	713	0.010101	0
21	515	0.010293618	1	576	0.010284	3	680	0.010145	3	712	0.010110	3	713	0.010101	0
22	515	0.010293618	2	575	0.010284	4	680	0.01013	3	712	0.010110	3	713	0.010101	0
23	513	0.010253643	3	575	0.010284	4	679	0.01013	3	712	0.010110	3	713	0.010101	0

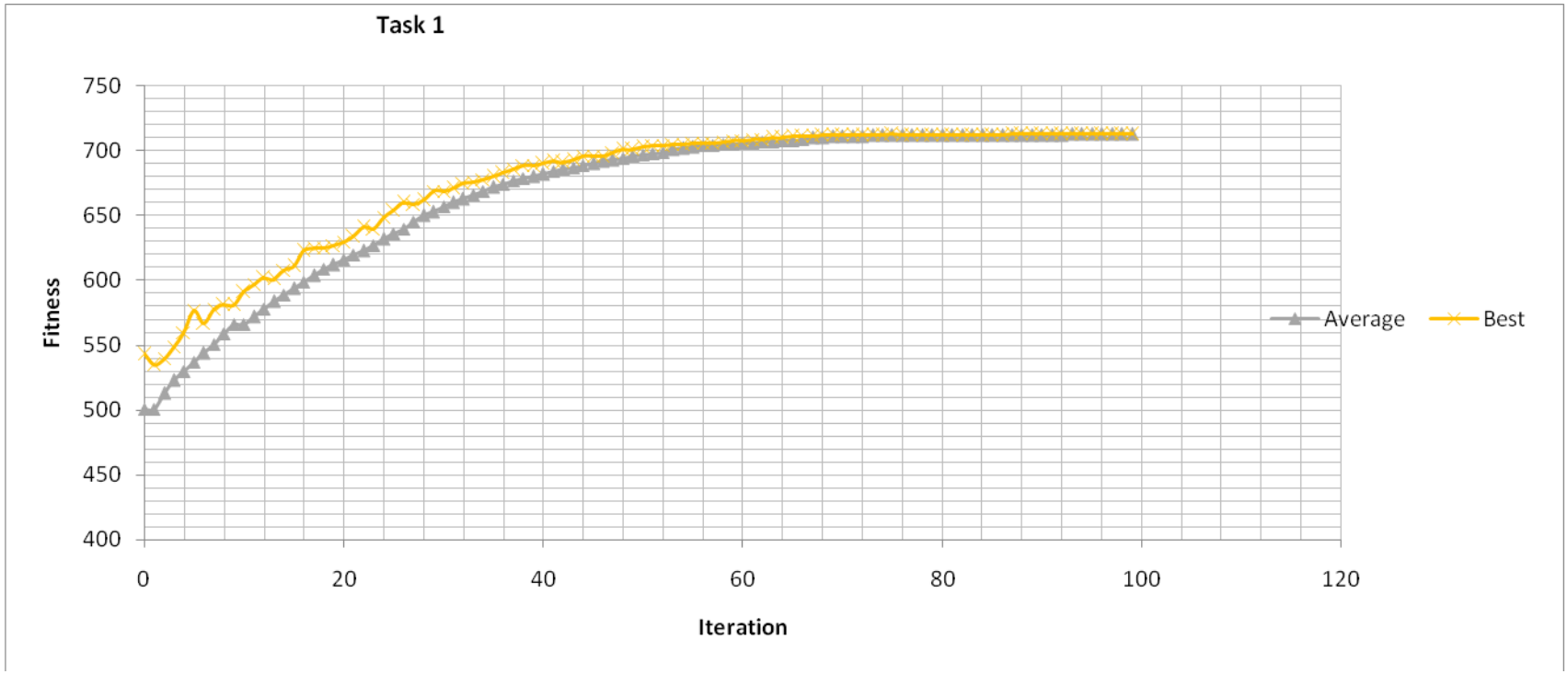
24	512	0.010233655	1	574	0.010284	4	679	0.01013	4	712	0.010110	3	713	0.010101	0
25	512	0.010233655	1	573	0.010284	1	679	0.01013	5	712	0.010110	3	713	0.010101	0
26	512	0.010233655	3	573	0.010266	0	679	0.01013	3	712	0.010110	3	713	0.010101	0
27	510	0.01019368	2	573	0.010266	2	679	0.01013	2	712	0.010110	3	713	0.010101	0
28	510	0.01019368	2	572	0.010266	2	679	0.01013	3	712	0.010110	2	713	0.010101	0
29	510	0.01019368	3	572	0.010248	22	679	0.01013	4	712	0.010110	2	713	0.010101	0
30	510	0.01019368	2	571	0.010248	3	679	0.01013	1	712	0.010110	1	713	0.010101	0
31	509	0.010173692	1	571	0.010248	1	679	0.01013	1	712	0.010110	1	713	0.010101	0
32	508	0.010153705	1	571	0.010248	1	679	0.01013	1	712	0.010110	1	713	0.010101	0
33	508	0.010153705	2	570	0.010231	1	679	0.01013	1	712	0.010110	2	713	0.010101	0
34	508	0.010153705	2	570	0.010231	0	679	0.01013	3	712	0.010110	1	713	0.010101	0
35	507	0.010133717	2	570	0.010231	0	679	0.01013	2	712	0.010110	1	713	0.010101	0
36	507	0.010133717	0	569	0.010231	0	679	0.01013	2	712	0.010110	1	713	0.010101	0
37	507	0.010133717	0	569	0.010231	0	679	0.01013	2	712	0.010110	2	713	0.010101	0
38	506	0.010113729	3	569	0.010213	0	679	0.01013	2	712	0.010110	2	713	0.010101	0
39	506	0.010113729	2	568	0.010213	3	679	0.01013	2	712	0.010110	0	713	0.010101	0
40	505	0.010093742	2	568	0.010213	1	679	0.01013	1	712	0.010110	0	713	0.010101	0
41	505	0.010093742	1	568	0.010213	1	679	0.01013	2	712	0.010110	2	713	0.010101	0

42	504	0.010073754	1	567	0.010213	1	679	0.01013	3	712	0.010110	2	713	0.010101	0
43	504	0.010073754	1	567	0.010213	1	679	0.01013	3	712	0.010110	2	713	0.010101	0
44	503	0.010053767	0	567	0.010213	3	679	0.01013	3	712	0.010110	2	713	0.010101	0
45	501	0.010013791	1	566	0.010213	2	679	0.01013	4	712	0.010110	2	713	0.010101	0
46	501	0.010013791	2	565	0.010213	2	679	0.01013	1	712	0.010110	2	713	0.010101	0
47	500	0.009993804	2	564	0.010195	2	679	0.010115	1	712	0.010110	1	713	0.010101	0
48	500	0.009993804	0	564	0.010195	2	678	0.010115	1	712	0.010110	2	713	0.010101	0
49	500	0.009993804	1	564	0.010195	2	678	0.010115	1	712	0.010110	2	713	0.010101	0
50	500	0.009993804	1	563	0.010195	0	678	0.01007	2	712	0.010110	2	713	0.010101	0
51	499	0.009973816	1	561	0.010195	0	675	0.01007	0	712	0.010110	2	713	0.010101	0
52	498	0.009953829	3	561	0.010195	0	675	0.01007	0	712	0.010110	2	713	0.010101	0
53	498	0.009953829	2	561	0.009999	0	675	0.01007	0	712	0.010110	2	713	0.010101	0
54	497	0.009933841	2	561	0.009999	0	675	0.01007	1	712	0.010110	2	713	0.010101	0
55	497	0.009933841	2	561	0.009999	0	675	0.01007	2	712	0.010110	2	713	0.010101	0
56	496	0.009913853	1	561	0.009999	1	675	0.01007	3	712	0.010110	2	713	0.010101	0
57	496	0.009913853	1	561	0.009999	1	675	0.01007	4	712	0.010110	2	713	0.010101	0
58	496	0.009913853	3	561	0.009999	1	675	0.01007	1	712	0.010110	1	713	0.010101	0
59	496	0.009913853	0	561	0.009999	2	675	0.01007	1	712	0.010110	1	713	0.010101	0

60	495	0.009893866	0	561	0.009999	2	675	0.01007	2	712	0.010110	2	713	0.010101	0
61	495	0.009893866	0	560	0.009999	2	675	0.01007	3	712	0.010110	1	713	0.010101	0
62	495	0.009893866	4	560	0.009999	1	675	0.01007	3	712	0.010110	1	713	0.010101	0
63	495	0.009893866	3	560	0.009981	1	675	0.01007	3	712	0.010110	3	713	0.010101	0
64	494	0.009873878	1	560	0.009981	1	675	0.01007	3	712	0.010110	2	713	0.010101	0
65	493	0.009853891	3	560	0.009981	1	675	0.01007	4	712	0.010110	1	713	0.010101	0
66	493	0.009853891	2	560	0.009981	1	675	0.01007	1	712	0.010110	1	713	0.010101	0
67	493	0.009853891	1	560	0.009981	3	675	0.01007	3	712	0.010110	1	713	0.010101	0
68	492	0.009833903	3	560	0.009981	3	675	0.01007	4	712	0.010110	2	713	0.010101	0
69	492	0.009833903	3	560	0.009981	3	675	0.01007	1	712	0.010110	1	713	0.010101	0
70	492	0.009833903	1	559	0.009981	1	675	0.01007	1	712	0.010110	1	713	0.010101	0
71	491	0.009813915	1	559	0.009981	1	675	0.01007	1	712	0.010110	2	713	0.010101	0
72	491	0.009813915	1	559	0.009963	1	675	0.01007	1	712	0.010110	2	713	0.010101	0
73	491	0.009813915	2	559	0.009963	0	675	0.01007	1	712	0.010110	2	713	0.010101	0
74	489	0.00977394	1	558	0.009963	0	675	0.01007	3	712	0.010110	2	713	0.010101	0
75	489	0.00977394	2	558	0.009963	0	675	0.01007	2	712	0.010110	2	713	0.010101	0
76	488	0.009753953	0	558	0.009945	2	675	0.01007	2	712	0.010110	0	713	0.010101	0

77	488	0.009753953	0	558	0.009945	2	675	0.01007	2	712	0.010110	0	713	0.010101	0
78	488	0.009753953	2	558	0.009945	2	675	0.01007	2	712	0.010110	1	713	0.010101	0
79	488	0.009753953	1	558	0.009945	2	675	0.01007	1	712	0.010110	1	713	0.010101	0
80	488	0.009753953	1	557	0.009945	2	675	0.010055	1	712	0.010110	2	713	0.010101	0
81	487	0.009733965	0	557	0.009945	3	674	0.010055	1	712	0.010110	1	713	0.010101	0
82	486	0.009713977	2	557	0.009928	0	674	0.010055	0	712	0.010110	2	713	0.010101	0
83	485	0.00969399	3	556	0.009928	0	674	0.010055	2	712	0.010110	1	713	0.010101	0
84	484	0.009674002	1	556	0.009928	0	674	0.010055	1	712	0.010110	1	713	0.010101	0
85	484	0.009674002	1	556	0.00991	0	674	0.010055	1	712	0.010110	1	713	0.010101	0
86	483	0.009654015	1	555	0.00991	0	674	0.010055	3	712	0.010110	1	713	0.010101	0
87	483	0.009654015	2	555	0.00991	1	674	0.010055	3	712	0.010110	0	713	0.010101	0
88	479	0.009574064	3	555	0.009892	1	674	0.010055	1	712	0.010110	0	713	0.010101	0
89	474	0.009474126	1	555	0.009892	1	674	0.010055	2	712	0.010110	0	713	0.010101	0
90	474	0.009474126	1	555	0.009892	2	674	0.010055	2	712	0.010110	0	713	0.010101	0
91	474	0.009474126	1	554	0.009892	2	674	0.010055	0	712	0.010110	0	713	0.010101	0
92	474	0.009474126	2	553	0.009892	3	674	0.010041	0	712	0.010110	0	713	0.010101	0
93	474	0.009474126	0	553	0.009874	4	673	0.010041	3	712	0.010110	0	713	0.010101	0
94	471	0.009414163	0	553	0.009856	4	673	0.010041	0	712	0.010110	0	713	0.010101	0

95	471	0.009414163	2	553	0.009856	0	673	0.010041	1	712	0.010110	0	713	0.010101	0
96	470	0.009394176	1	553	0.009856	0	673	0.010041	2	712	0.010110	0	713	0.010101	0
97	469	0.009374188	1	553	0.009856	0	673	0.010041	2	712	0.010110	0	713	0.010101	0
98	468	0.0093542	2	552	0.009856	1	673	0.010041	3	712	0.010110	0	713	0.010101	0
99	458	0.009154324	0	552	0.009856	1	673	0.010032	1	712	0.010110	0	713	0.010101	0
Average	500			552			677			712			713		
Best	541			94			683			713			713		
SUM		1		56651	1			1			1			1	



The best fitness chromosome in the 1st generation is 541, in the 10th – 552, and in the 100th - generation 713.

Since 90 iteration, there is no improvement of chromosomes because we achieved the best result after crossing all possible parents!

```
    (() => {
"use strict";
const populationCount = 100,genSize = 1000,itterationsCount = 100;
let generation = generateGeneration(populationCount, genSize),itterationNum = 0;
for (let i = 0; ++i < itterationsCount;)
 {
	generation = generation.sort((gen1, gen2)
	{
		return calcOnes(gen2) - calcOnes(gen1);
	});
	logGeneration(generation);
	generation = nextGeneration(generation);
}
logGeneration(generation);
return;
function generateGeneration(count, size)
 {
	let generation = [];
	for (let i = -1; ++i < count;)
	 {
		let gen = [];
		for (let j = -1; ++j < size;)
		{
			gen.push(getRandomBinary());
		}
	generation.push(gen);
	}
return generation;
}
function calcOnes(gen)
{
	let count = 0;
	for (let i = -1; ++i < gen.length;)
		 if (gen[i]) count++;
	return count;
}
function nextGeneration(parentGeneration)
{
	let repeatCount = populationCount / 2,
	newGeneration = [];
	for (let i = 0; ++i < populationCount;) {
		let firstIndex = getRandomInt(0, populationCount),
```

```
                secondIndex = getRandomInt(0, populationCount);
                  document.write(firstIndex + " " + secondIndex + " | ");
                newGeneration.push(...crossover(parentGeneration[firstIndex],
                parentGeneration[secondIndex]));
        }
return newGeneration;
}
function crossover(firstParent, secondParent)
{
        let crossIndex = getRandomInt(1, firstParent.length);
        let firstPartFirst = firstParent.slice(0, crossIndex),secondPartFirst = firstParent.slice(crossIndex, firstParent.length),firstPartSecond = secondParent.slice(0, crossIndex),secondPartSecond =
secondParent.slice(crossIndex, secondParent.length);
        return [firstPartFirst.concat(secondPartSecond),
        firstPartSecond.concat(secondPartFirst)];
}
function getRandomBinary(min, max) {
        return Math.round(Math.random());
}
function getRandomInt(min, max) {
        return Math.floor(Math.random() * (max - min)) + min;
}
function bestCh (generation) {
let best = calcOnes(generation[0]); ;
        for (let i = 1; i < generation.length; i++) {
                if (calcOnes(generation[i])>best) best = calcOnes(generation[i]);
        }
return best;
}
function logGeneration(generation) {
        let average = 0;
        let allsum=0;
        let onlyOnes=0;
        let    onlyOnesCh=0;
        for (let i = 0; i < generation.length; i++) allsum +=calcOnes(generation[i]);
                 for (let i = 0; i < generation.length; i++)
                 {
                         onlyOnes=calcOnes(generation[i]);
                            onlyOnesCh=onlyOnes/allsum;
                            document.write(onlyOnes+"|"+ onlyOnesCh + '<br>');
                }

        average = Math.floor(allsum / generation.length);
        document.write("Average "+average + '<br>');
        document.write("Best" + bestCh(generation)+'<br>');
```

```
}
    })();
```