Moad Hani (moad.hani.001@student.uni.lu) - 0190054306

Problem Solving Project: Evolutionary Reconstruction of a shredded picture

Pascal Bouvry, Emmanuel Kieffer

1 Objectives

Introduce a meta-heuristic framework to students.

Familiarize students with the concept of Genetic Algorithm through a hand-on experience.

2 Introduction

In this project, students will be tasked with restoring the provided shredded picture. The restoration is to be done through the use of Genetic Algorithm (GA). The essential functions for image processing and GA structure are provided in Python. Students are encouraged to study the docu-ment of "Distributed Evolutionary Algorithms (DEAP)", a meta-heuristic framework [1] since some parts of the code need to be implemented and/or modified, to achieve the result.

In order to restore a shredded documents, one need to put document strands in a correct order as can be seen in the Fig.1. In this project, students are provided with a digital image which is supposedly shredded by shuffling the order of column in the picture. As an end result, the shredded pictures should be restored.

3 Project Description

In this problem, students will be asked to get familiar with a python evolutionary computing library which provide all necessary tools to implement a genetic algorithm. The library offers most of the basic implementations, but students are still required to implements their own functions to make it suit this problem. The DEAP library is very well documented and provide numerous examples. The goal of this project is to make use of the notions learned during this semester as well as taking advantage of the existing tools to solve optimization problems.

The task is to reconstruct a picture that has been shredded. In fact, the columns of the original image have been shuffled so that the objects are not recognizable. To reconstruct the real order, you will have to implement a python genetic algorithm using the DEAP library. In order to evaluate your solution, you will have to your disposal an "oracle" which is a black box function. When you wish to evaluate your solution, you have to provide a permutation list of the columns to the oracle which will give you back a score based on how close the restored picture resemble the original. The closer to 0 the better. The oracle can also be used to display the restored picture. An example is shown in Fig. 2 and Fig. 3 [2].

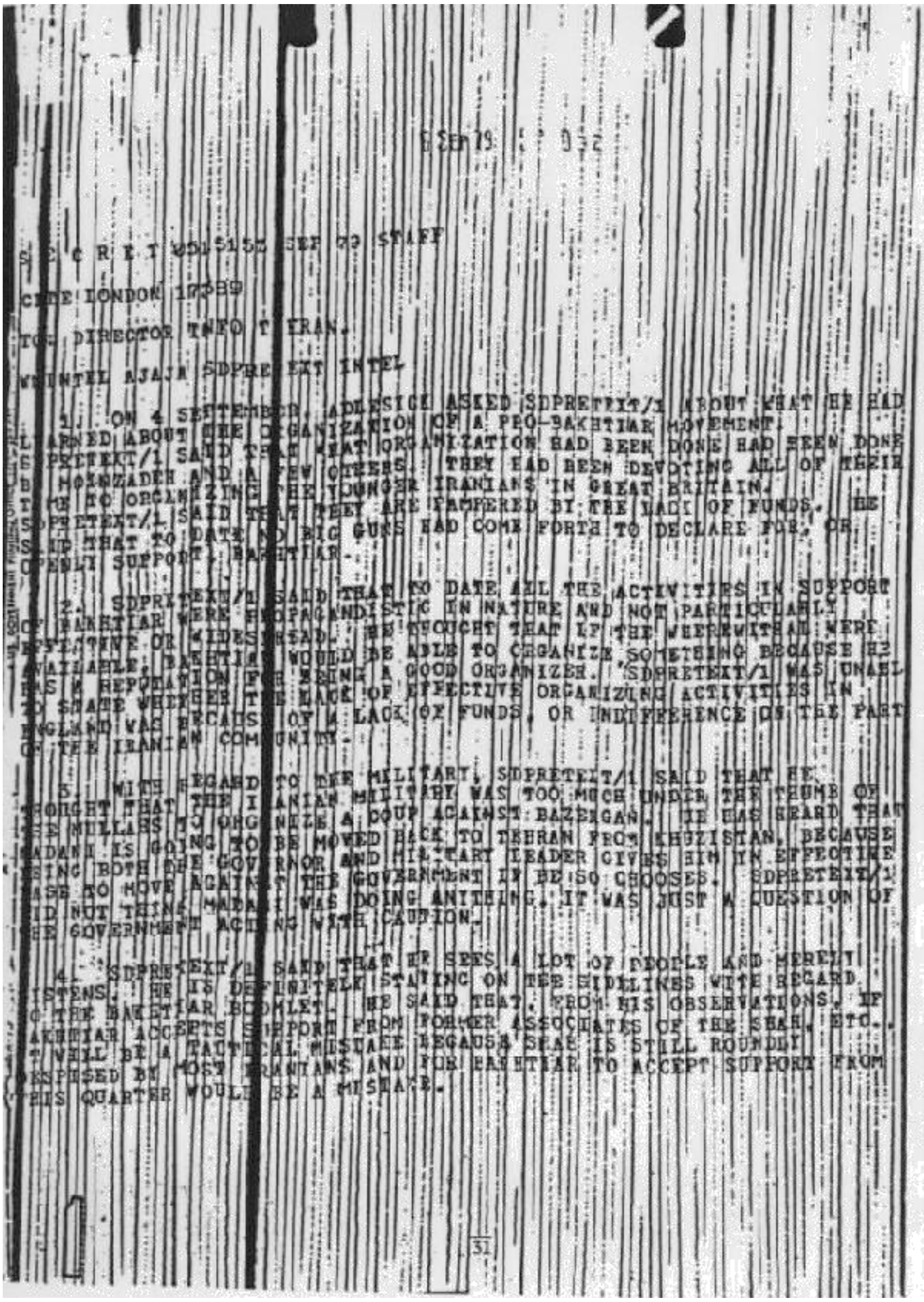
4 Required Tools

The provided source code and DEAP library are implemented in Python. There are also dependen-cies that need to be installed for the provided source code to work. Since the code is implemented is Python, they are generally cross-platform. In this section, package installation instructions are given.

Programming language: Python 2 or 3

[1] http://deap.readthedocs.io/en/master/

[2] both pictures are taken from https://github.com/robinhouston/image-unshredding.



6 SEP 79 [illegible] 032

S E C R E T [illegible] SEP 79 STAFF

CITE LONDON 17389

TO: DIRECTOR INFO TEHRAN.

WNINTEL AJAJA SDPRETEXT INTEL

1. ON 4 SEPTEMBER, ADDISICK ASKED SDPRETEXT/1 ABOUT WHAT HE HAD LEARNED ABOUT THE ORGANIZATION OF A PRO-BAKHTIAR MOVEMENT. SDPRETEXT/1 SAID THAT WHAT ORGANIZATION HAD BEEN DONE HAD BEEN DONE BY HOSNZADEH AND A FEW OTHERS. THEY HAD BEEN DEVOTING ALL OF THEIR TIME TO ORGANIZING THE YOUNGER IRANIANS IN GREAT BRITAIN. SDPRETEXT/1 SAID THAT THEY ARE HAMPERED BY THE LACK OF FUNDS. HE SAID THAT TO DATE NO BIG GUNS HAD COME FORTH TO DECLARE FOR, OR OPENLY SUPPORT, BAKHTIAR.

2. SDPRETEXT/1 SAID THAT TO DATE ALL THE ACTIVITIES IN SUPPORT OF BAKHTIAR WERE PROPAGANDISTIC IN NATURE AND NOT PARTICULARLY EFFECTIVE OR WIDESPREAD. HE THOUGHT THAT IF THE WHEREWITHAL WERE AVAILABLE, BAKHTIAR WOULD BE ABLE TO ORGANIZE SOMETHING BECAUSE HE HAS A REPUTATION FOR BEING A GOOD ORGANIZER. SDPRETEXT/1 WAS UNABLE TO STATE WHETHER THE LACK OF EFFECTIVE ORGANIZING ACTIVITIES IN ENGLAND WAS BECAUSE OF A LACK OF FUNDS, OR INDIFFERENCE ON THE PART OF THE IRANIAN COMMUNITY.

3. WITH REGARD TO THE MILITARY, SDPRETEXT/1 SAID THAT HE THOUGHT THAT THE IRANIAN MILITARY WAS TOO MUCH UNDER THE THUMB OF THE MULLAHS TO ORGANIZE A COUP AGAINST BAZERGAN. HE HAS HEARD THAT MADANI IS GOING TO BE MOVED BACK TO TEHRAN FROM KHUZISTAN, BECAUSE BEING BOTH THE GOVERNOR AND MILITARY LEADER GIVES HIM AN EFFECTIVE BASE TO MOVE AGAINST THE GOVERNMENT IF HE SO CHOOSES. SDPRETEXT/1 DID NOT THINK MADANI WAS DOING ANYTHING. IT WAS JUST A QUESTION OF THE GOVERNMENT ACTING WITH CAUTION.

4. SDPRETEXT/1 SAID THAT HE SEES A LOT OF PEOPLE AND MERELY LISTENS. HE IS DEFINITELY STAYING ON THE SIDELINES WITH REGARD TO THE BAKHTIAR BOOKLET. HE SAID THAT, FROM HIS OBSERVATIONS, IF BAKHTIAR ACCEPTS SUPPORT FROM FORMER ASSOCIATES OF THE SHAH, ETC., IT WILL BE A TACTICAL MISTAKE BECAUSE SHAH IS STILL ROUNDLY DESPISED BY MOST IRANIANS AND FOR BAKHTIAR TO ACCEPT SUPPORT FROM THIS QUARTER WOULD BE A MISTAKE.

31

Figure 1: A reconstructed shredded document

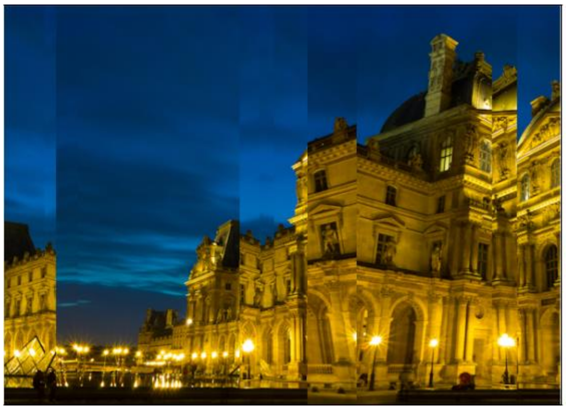


Figure 2: A Shredded picture

Figure 3: A restored picture

Evolutionary Computing library: Distributed Evolutionary Algorithms (DEAP) library

If you do not have already the pip package manager, go on this page https://pip.pypa.io/en/stable/installing/ and download get-pip.py file. In a terminal, install pip by entering the command python get-pip.py.

Once pip has been installed, execute all the following commands:

pip install numpy

pip install Image

pip install deap

5 How to use the Oracle?

The oracle is a python binary module named blackbox.py. Fig. 4 shows how to communicate with the oracle.

Figure 4: An example of how to use the provided Oracle.

You oracle module should always be placed in the same directory as your code. In addition, the file original_picture:png:enc should be also placed in the same directory.

The oracle has only two methods and only accept a permutation list of integers. The size of any permutation list should 128.

show_solution(permutation_list) which display a picture after applying the permutation

evaluate_solution(permutation_list; save_file_name) which provide a score that you want to minimize

Your genetic algorithm should theoretically employed only the method evaluate_solution(permutation_list)" in order to evaluate solution. Once the algorithm stops and give you the best found solution, you can call the show_solution(permutation_list; save_file_name) method to visualize or save the result.

6 Report

Along with the implementation, students are required to turn in a report that contains details of their work. The guidelines are given in this section.

Problem Solving Project: Evolutionary Reconstruction of a shredded picture

Moad Hani (moad.hani.001@student.uni.lu)

1. Which encoding is relevant for this problem?

Provide the name and the encoding, your are going to use for this problem. Describe it by providing an example

At a high level, the community calls the genetic encoding the genotype and the genes' ultimate form or behavior in the solution the phenotype. The genotype is the way the parts of the problem are encoded so they can be manipulated by the genetic algorithm and/or engine. **The real-coded GA** is most suitable for optimization in a continuous search space. Uses the direct representations of the design paparmeters. Thus, avoids any intermediate encoding and decoding steps.

Genotype :

x	**y**

Phenotype :

5.28	**-475.36**

Real-value representation

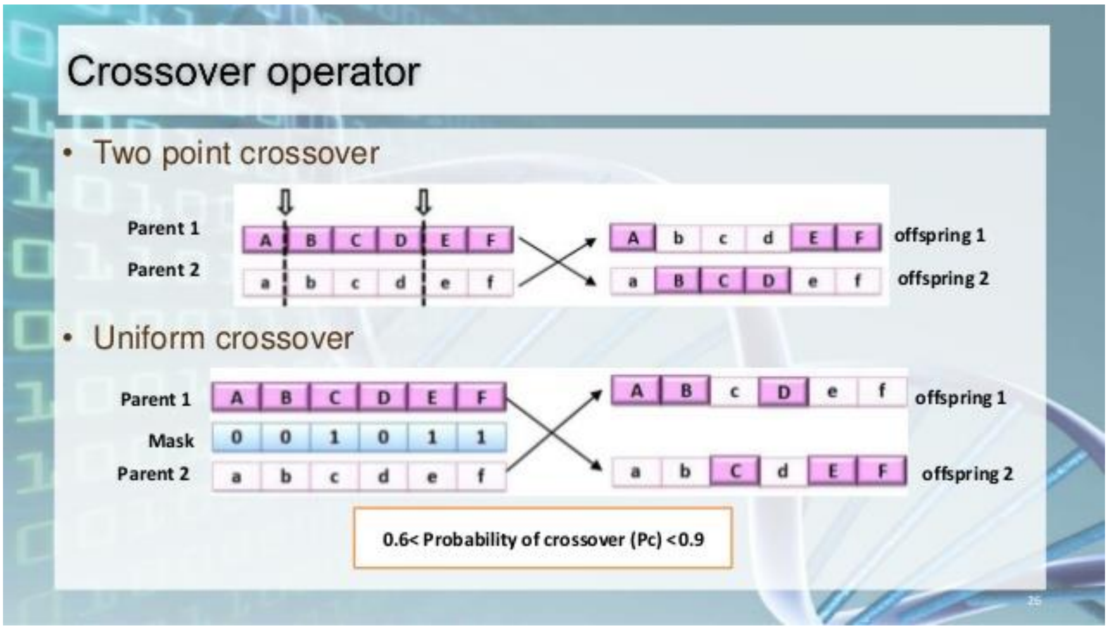
2. Which cross-over operator are you going to use?

Provide its name. Describe it by providing an example (e.g. a picture) to demonstrate how it works. Does it conserve the permutation property?

deap.tools.**cxPartialyMatched**(*ind1*, *ind2*)

Executes a partially matched crossover (PMX) on the input individuals. The two individuals are modified in place. This crossover expects sequence individuals of indices, the result for any other type of individuals is unpredictable.

Parameters:	**ind1** – The first individual participating in the crossover. **ind2** – The second individual participating in the crossover.
Returns:	A tuple of two individuals.



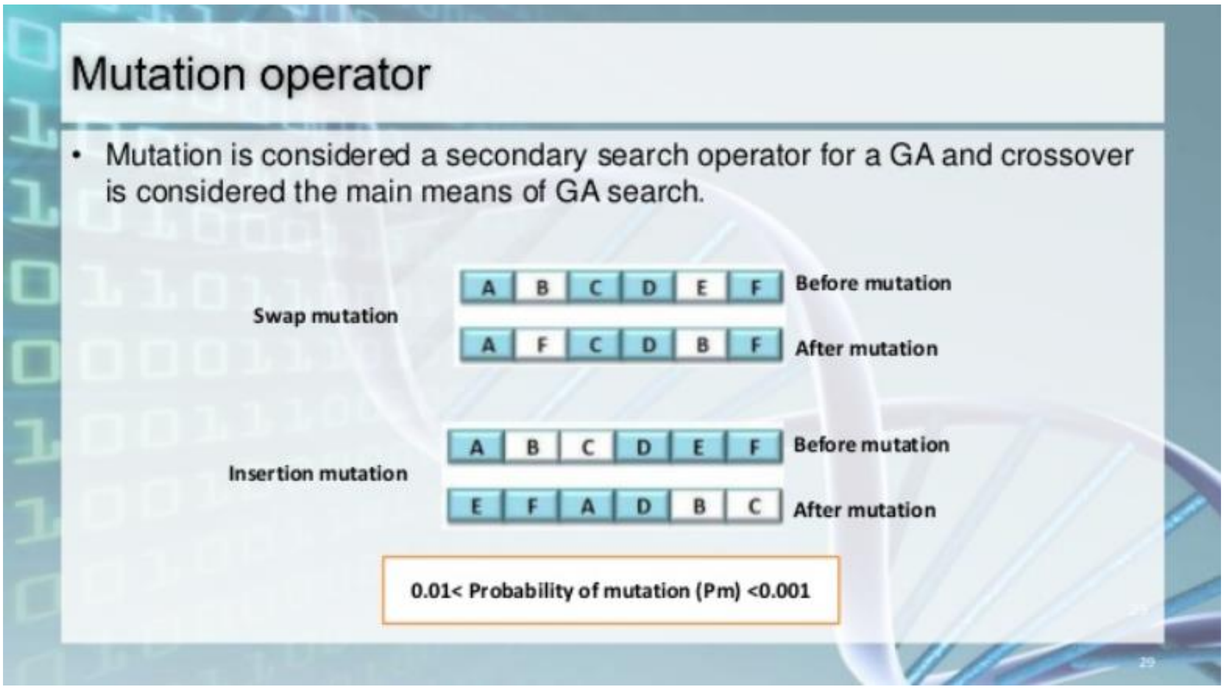
3. Which mutation operator are you going to use?

Provide its name. Describe it by providing an example (e.g. a picture) to demonstrate how it works. Does it conserve the permutation property?

deap.tools.**mutShuffleIndexes**(*individual*, *indpb*)

Shuffle the attributes of the input individual and return the mutant. The *individual* is expected to be a sequence. The *indpb* argument is the probability of each attribute to be moved. Usually this mutation is applied on vector of indices.

Parameters:	**individual** – Individual to be mutated. **indpb** – Independent probability for each attribute to be exchanged to another position.
Returns:	A tuple of one individual.



4. Which selection operator are you going to use?

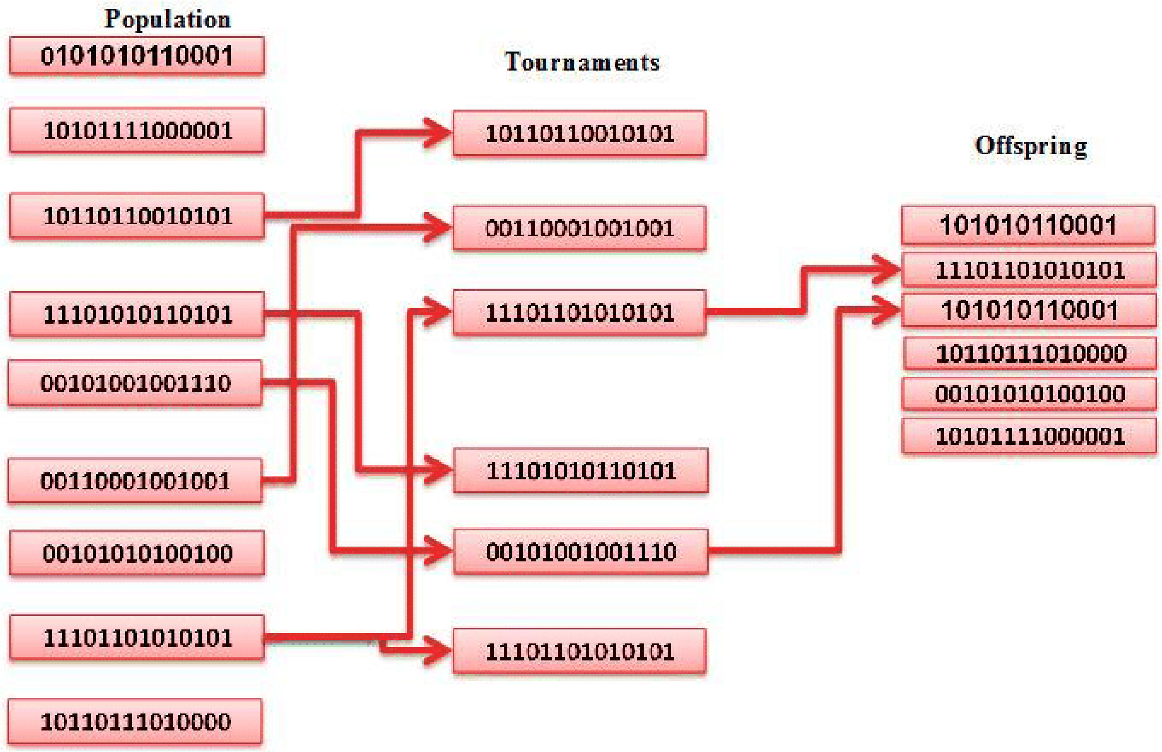
Provide its name. Describe it by providing an example (e.g. a picture) to demonstrate how it works. Does it conserve the permutation property?

deap.tools.**selTournament**(*individuals*, *k*, *tournsize*, *fit_attr='fitness'*)

Select the best individual among *tournsize* randomly chosen individuals, *k* times. The list returned contains references to the input *individuals*.

Parameters:	**individuals** – A list of individuals to select from. **k** – The number of individuals to select. **tournsize** – The number of individuals participating in each tournament. **fit_attr** – The attribute of individuals to use as selection criterion
Returns:	A list of selected individuals.

This function uses the **choice()** function from the python base **random** module.


Population
0101010110001
10101111000001
10110110010101
11101010110101
00101001001110
00110001001001
00101010100100
11101101010101
10110111010000
Tournaments
10110110010101
00110001001001
11101101010101
11101010110101
00101001001110
11101101010101
Offspring
101010110001
11101101010101
101010110001
10110111010000
00101010100100
10101111000001

5. Implementation

Use the genetic algorithm implementation provided by DEAP using the operators and the parame-ters described above. You have to provide a work-flow diagram to describe how it works. Use the following parameters:

1. population size (n): 100
2. generations (ngen): 500
3. cross-over probability (cxpb): 0.9
4. allele mutation probability (indpb) : 0.005
5. mutation probability (mutpb) : 1
6. Additional parameters are left to your discretion (e.g. tournament selection, ...)

6. Experimental Result

Genetic Algorithm (GA) is based on stochastic mechanism, numbers of run need to be performed to obtain a good statistical confidence. You will be asked to perform 30 runs. The best solution of each of the 30 expected runs has to be recorded and all runs should be described in Tab. 1.

In addition, the report should provide average convergence curve. The curve is plot from the average of each generation best results (i.e minimum fitness) from each runs. A record example can be found in Tab. 2. A curve from the table is shown in Fig. 5

Finally, the result picture (best fitness score) must be included in the report whether it is com-pletely restored or partially restored. You can export it by applying the "show_solution(permutation_list,

Table 1: An experimental result table

No. Run	Fitness Value
1	1337088
2	1816064
3	3067648
4	2247936
5	2489856
6	3929344
7	3519744
8	4366592
9	3203840
10	3251456
11	2257408
12	1469696
13	2075392
14	2487040
15	1221632
16	3975936
17	884480
18	3437568
19	2414848
20	1074176
21	3015424
22	3014144
23	2191360
24	1220352
25	3168256
26	2243072
27	3260672
28	1739776
29	1768704
30	1765513
Average Fitness	2463833.9
Maximum Fitness	4366592
Minimum Fitness	884480
Standard Deviation	9.20917 e+05

Standard Deviation, σ: 920917.70983841

Count, N: 30
Sum, Σx: 73915017
Mean, μ: 2463833.9
Variance, σ^2: 848089428294.02

Steps

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}.$$

$$\sigma^2 = \frac{\Sigma(x_i - \mu)^2}{N}$$

$$= \frac{(1337088 - 2463833.9)^2 + ... + (1765513 - 2463833.9)^2}{30}$$

$$= \frac{25442682848821}{30}$$

$$= 848089428294.02$$

$$\sigma = \sqrt{848089428294.02}$$

$$= 920917.70983841$$

Table 2: A record of population statistics during 100 generations

Gen	min fitness	max fitness	avg fitness	std deviation
1	37526528.0	39301888.0	38408537.6	407645.73532488436
2	37284096.0	39341568.0	38264934.4	446969.2109588893
3	36815872.0	39136512.0	38081047.04	438668.3016630191
4	37007872.0	39116288.0	37966128.64	449031.5580368943
5	36706816.0	39284736.0	37870894.08	500778.52258605126
6	36399872.0	39009792.0	37767782.4	523176.6668506631
7	36505600.0	39031552.0	37776104.96	534788.2712853396
8	36489216.0	38821632.0	37667020.8	489164.5297758085
9	36343040.0	38853632.0	37656609.28	545777.7031864497
10	36178688.0	38962688.0	37568867.84	599492.7408245979
11	36060416.0	38738944.0	37414789.12	550372.2373777106
12	36015872.0	38924800.0	37375626.24	637247.3227745218
13	35457536.0	38859520.0	37298145.28	623067.4986764035
14	35457536.0	38975232.0	37330705.92	786365.892217268
15	35492864.0	38457344.0	37089195.52	607570.8829967614
16	35544320.0	38467328.0	36887342.08	573037.516912361
17	35727104.0	38722048.0	36826795.52	571390.6283521424
18	34956288.0	38670336.0	36688983.04	655094.5444402295
19	34907904.0	38117376.0	36464494.08	648029.505863343
20	34230528.0	37599232.0	36170690.56	570734.8651618186
21	34230528.0	37572352.0	35900940.8	680980.6023465
22	34110720.0	37853696.0	35804142.08	636729.1605622414
23	34110720.0	37700608.0	35697523.2	651637.6930549209
24	34356224.0	37319168.0	35595563.52	641107.1170098303
25	34476032.0	38034688.0	35572771.84	711569.9697412406
26	33326336.0	36619264.0	35220625.92	611548.8217376604
27	33588480.0	37526016.0	35185738.24	711041.8740304611
28	33246208.0	36686336.0	34961738.24	647664.2079742025

29	33246208.0	36442368.0	34753093.12	698549.6555520911
30	33209088.0	35970048.0	34404971.52	656242.6936178395
31	32819200.0	35739648.0	34277752.32	597586.348996568
32	32787456.0	35955200.0	34093493.76	631851.5131109112
33	32872704.0	35093504.0	33862100.48	491848.09528418054
34	32690944.0	35317760.0	33862868.48	542143.0530349254
35	32480512.0	35187200.0	33741491.2	581088.774633872
36	32671232.0	34848000.0	33701073.92	561319.6008709592
37	32489216.0	34619904.0	33541050.88	515544.3886907223
38	32163584.0	34797824.0	33329850.88	494440.43750750384
39	32113152.0	34204928.0	33129205.76	468616.5260441842
40	32103680.0	34557184.0	33082362.88	512745.76020435407
41	32031232.0	34524416.0	32994508.8	520553.93374040123
42	31631360.0	34375680.0	33021038.08	561250.5232451035
43	31638272.0	34780672.0	32904291.84	637724.7255273259
44	31556352.0	34742784.0	32840670.72	599311.9260207227
45	31725056.0	34267904.0	32815406.08	606981.808480232
46	31725056.0	34169344.0	32747013.12	544074.1786377934
47	31515136.0	34691840.0	32731668.48	627412.501185728
48	31582464.0	34512640.0	32642577.92	588937.2911228709
49	31559424.0	34402304.0	32689638.4	574096.4156431417
50	31330816.0	33874688.0	32562449.92	580518.1929069172
51	30389504.0	33952256.0	32386728.96	615170.3402029495
52	30357504.0	33351680.0	32143011.84	493534.8246799336
53	30604288.0	33541120.0	31995422.72	574421.5982824595
54	30349312.0	33046272.0	31865249.28	574548.8069689608
55	30355456.0	33262848.0	31667409.92	514383.8965412145
56	30612736.0	32637952.0	31625756.16	509715.7817525079

57	30322432.0	32726784.0	31440760.32	453329.2710176918
58	30292480.0	32890880.0	31415283.2	509387.4151836445
59	30087936.0	32497920.0	31320826.88	503591.3075653505
60	30093312.0	32752384.0	31226324.48	537415.5215854668
61	29818624.0	32186624.0	31013908.48	497034.28246740537
62	29673472.0	31993600.0	30913090.56	486563.41190514324
63	29889024.0	31926272.0	30885463.04	488279.82955268427
64	29550848.0	32389632.0	30731491.84	560241.7885506569
65	29452288.0	32123648.0	30587005.44	591007.542622848
66	29278208.0	32028928.0	30598568.96	568776.5796508427
67	28552448.0	31927552.0	30554283.52	668492.1240501042
68	29188864.0	32628736.0	30387264.0	654196.4078033972
69	28867584.0	31833600.0	30136110.08	540491.2025597068
70	28847616.0	32093952.0	30043292.16	609000.2544150788
71	28844032.0	31305472.0	29937320.96	571080.67863101
72	28538880.0	31166464.0	29735470.08	565749.8230689418
73	28393984.0	30722560.0	29492743.68	502929.93691699335
74	28426240.0	31201280.0	29415577.6	526624.8615036873
75	28087808.0	30654720.0	29322329.6	509273.70783725247
76	28000512.0	31436032.0	29236462.08	542734.4175008932
77	28159744.0	31131904.0	29193287.68	570528.3286460844
78	27629568.0	30724864.0	29076321.28	602405.0100272916
79	27629568.0	30472960.0	28975165.44	603644.4974783799
80	27576576.0	30043136.0	28900390.4	530489.3604959043
81	27254784.0	30743552.0	28827624.96	616970.9432296592
82	27086336.0	30309376.0	28780229.12	595666.7322587894
83	27092480.0	30075136.0	28731937.28	619229.2103576211
84	27013376.0	30886912.0	28766684.16	675600.1631571698
85	27269632.0	30931968.0	28691576.32	648894.2637934614
86	27356416.0	30098688.0	28510883.84	562455.953720979

87	27186432.0	30688512.0	28495011.84	665598.3793807776
88	26562048.0	29752832.0	28323804.16	649196.8036112975
89	26562048.0	30422272.0	28139095.04	709843.8177624384
90	26562048.0	30075392.0	28008629.76	771948.7860469606
91	26504448.0	29273600.0	27607831.04	631180.5602150725
92	26505472.0	29731584.0	27439124.48	597801.8708156038
93	26279168.0	28697856.0	27199951.36	534284.4957274143
94	26208768.0	28423168.0	26978621.44	379844.5330048376
95	26093568.0	28561152.0	26911086.08	422197.95264000003
96	25972224.0	28110592.0	26794718.72	381945.2105963584
97	25396992.0	28109568.0	26738864.64	459127.2530103157
98	25281792.0	27844352.0	26589527.04	437315.1699661743
99	25009152.0	28566272.0	26458122.24	572312.1768275363
100	25009152.0	28260096.0	26339189.76	514887.03345677827

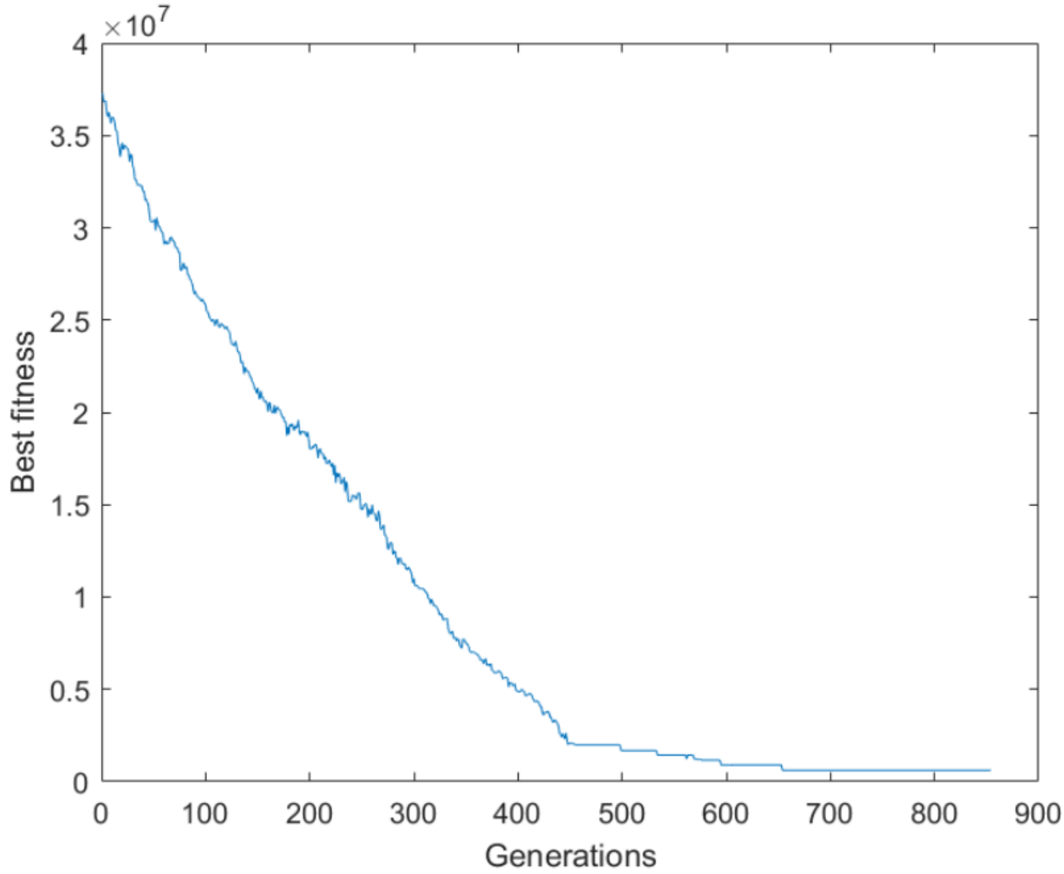
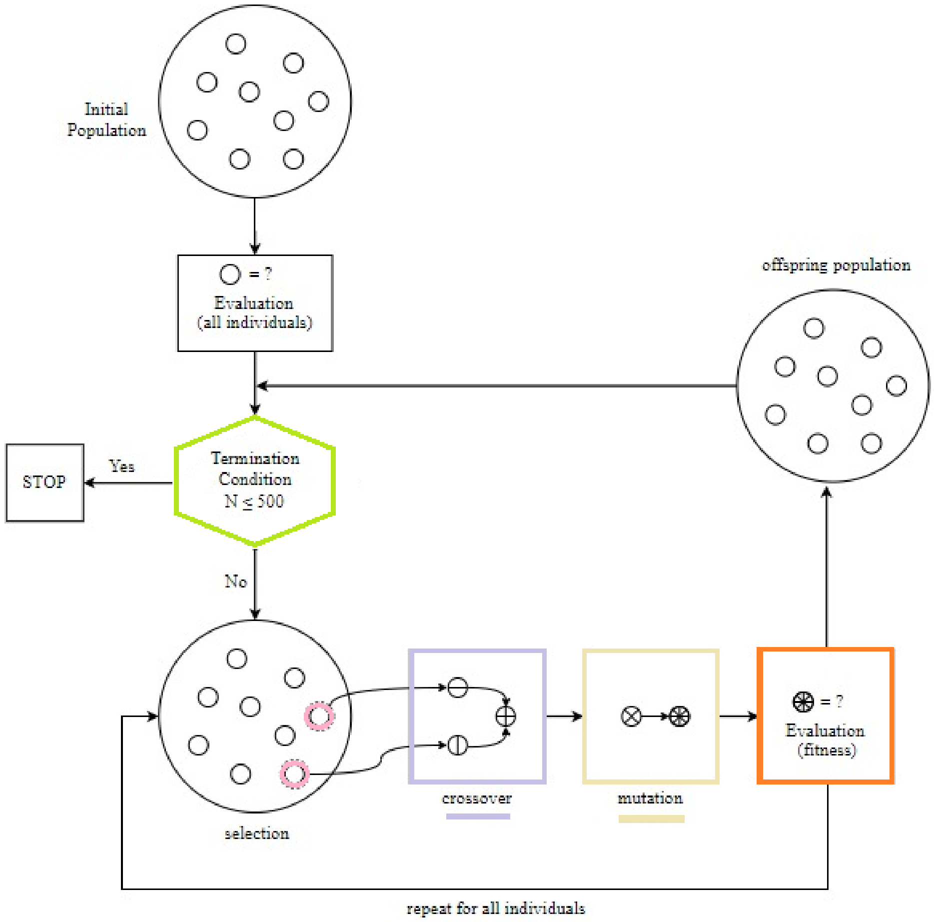


Figure 5: Example of convergence curve after 850 generations

Original picture

Reconstructed picture

Workflow diagram :



Conclusion:

A gene is the GA's representation of a single factor (i.e. a design parameter), which has a domain of values (continuous, discontinuous, discrete etc.) symbol, numbering etc.

In GA, there is a mapping from genotype to phenotype. This eventually decideds the performance (namely speed and accuracy) of the problem solving.

Permutation property has been recognized as a common but challenging feature in combinatorial problems. Because of their complexity, recent research has turned to genetic algorithms to address such problems. Although genetic algorithms have been proven to facilitate the entire space search, they lack in fine-tuning capability for obtaining the global optimum.

Genetic algorithms and genetic programming are very good at finding solutions to very large problems. They do it by taking millions of samples from the search space, making small changes, possibly recombining parts of the best solutions, comparing the resultant fitness against that of the current best solution, and keeping the better of the two. This process
repeats until a stop condition like one of the following occurs: the known solution is found, a solution meeting all requirements is found, a certain number of generations has passed, a specific amount of time has passed, etc.