Genetic Algorithm Based on Natural Selection Theory (GABONST)

Chinmaya Srivatsa
Tse-Han Pan

Outline

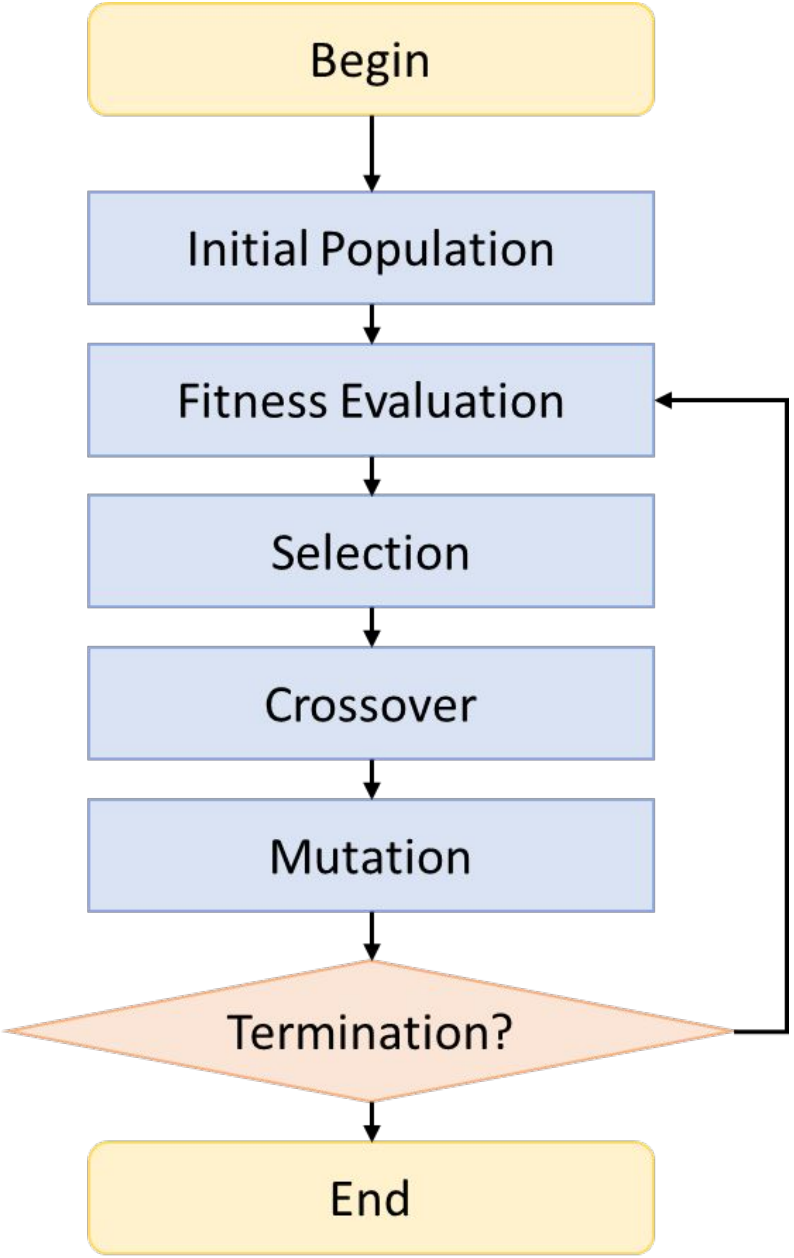
- Project Objective
- Algorithm Recap
- Implementations
- Evaluations
- TSP Art
- Discussions

Project Objective

- Implement Genetic Algorithm Based on Natural Selection Theory (GABONST) in Julia for TSP optimization
- Benchmark and present the results with graphs and statistics
- Generate TSP art with GABONST

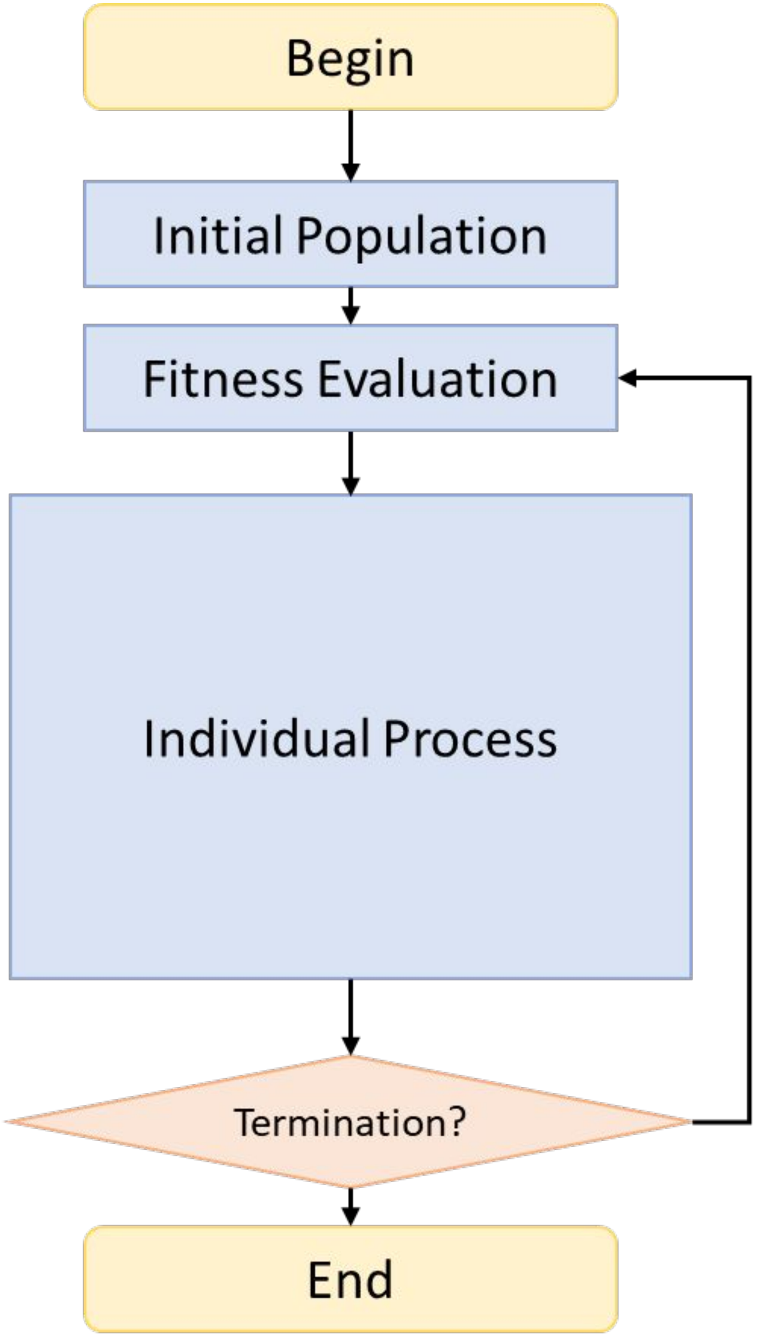
Algorithm Recap

- Recap of Genetic Algorithm

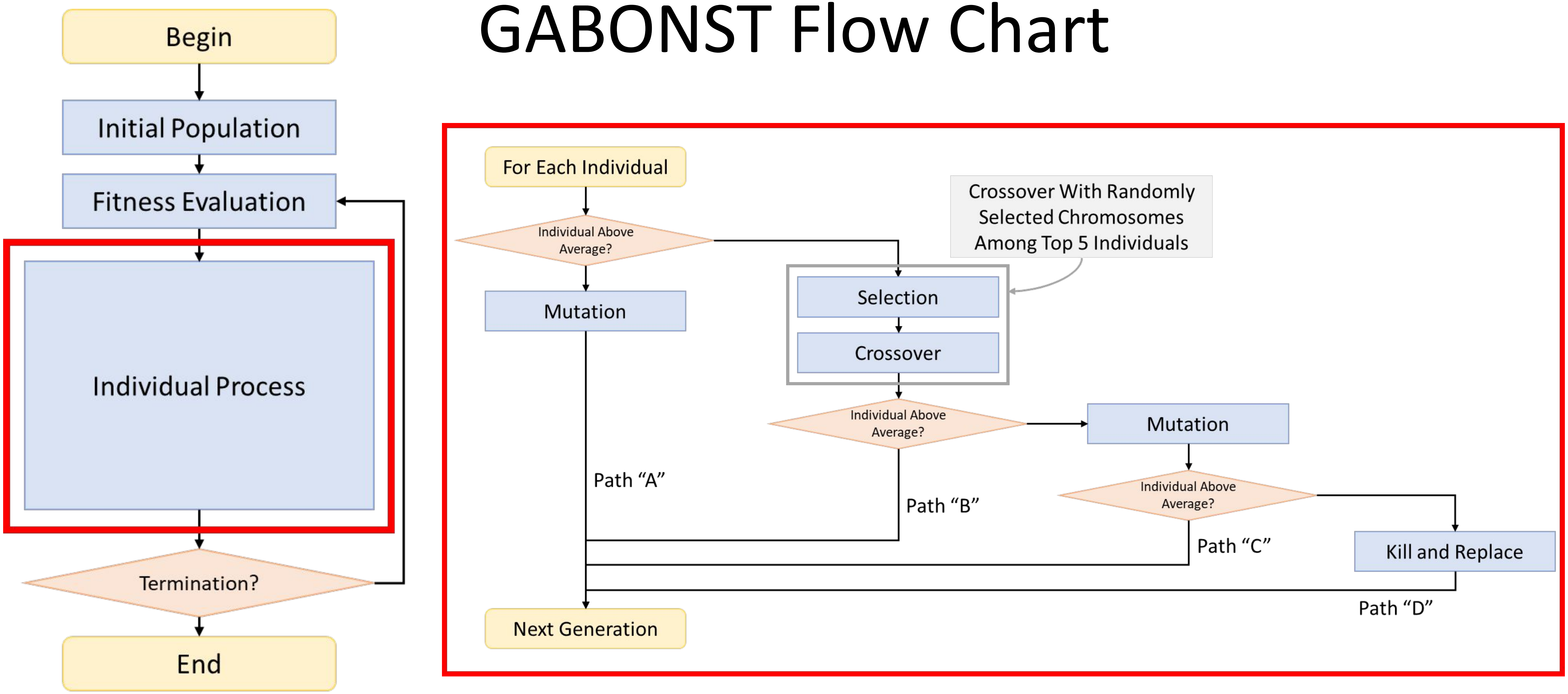


Algorithm Recap

- GABONST overview



GABONST Flow Chart



Implementations

- GABONST implementation in the paper
  - Arithmetic Crossover

Each alpha = Uniform(-gamma, gamma + 1), gamma = 0.4 in this case

Alpha Vector	**0.3**	**0.8**	**-0.1**	**0.7**	**-0.2**	**1.3**
Chromosome A	**0.4**	**0.2**	**-0.8**	**-0.3**	**-0.1**	**0.9**
Chromosome B	**0.2**	**0.5**	**0.7**	**0.4**	**0.8**	**-0.7**

Crossed value = alpha*A + (1-alpha)*B, Values are capped at limits, [-1, 1] in this case

Crossed Value	**0.26**	**0.26**	**0.85**	**-0.09**	**0.98**	**1**

Capped

Implementations

- GABONST implementation in the paper
  - Uniform Mutation

Uniform(0, 1), mutate if value less than mutation rate, mutation rate = 0.25 in this case

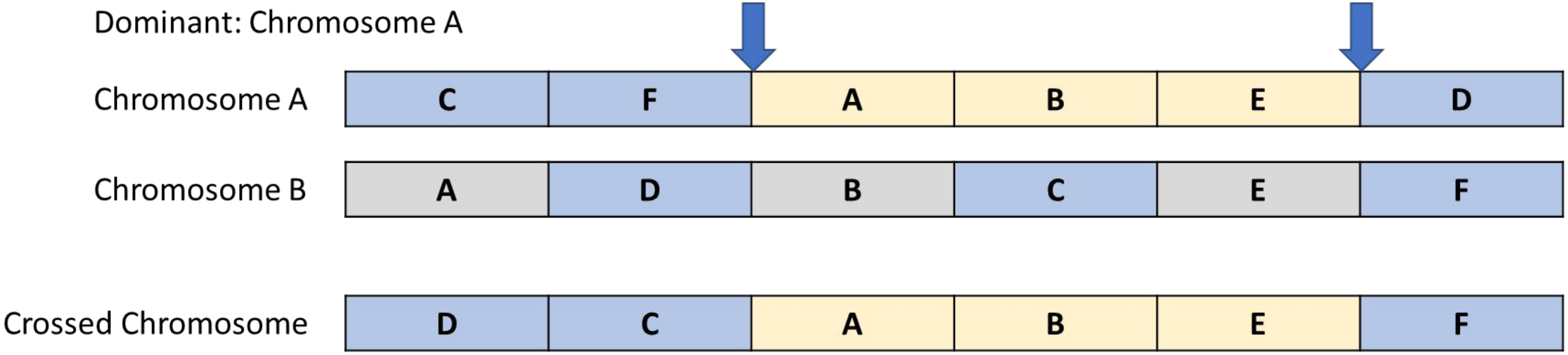
Mutation Vector	**0.3**	**0.8**	**0.1**	**0.7**	**0.2**	**0.3**
Chromosome A	**0.4**	**0.2**	**-0.8**	**-0.3**	**-0.1**	**0.9**

Mutated value = Uniform(lower_bound, upper_bound), bounds = [-1, 1] in this case

Mutated A	**0.4**	**0.2**	**0.3**	**-0.3**	**-0.6**	**0.9**

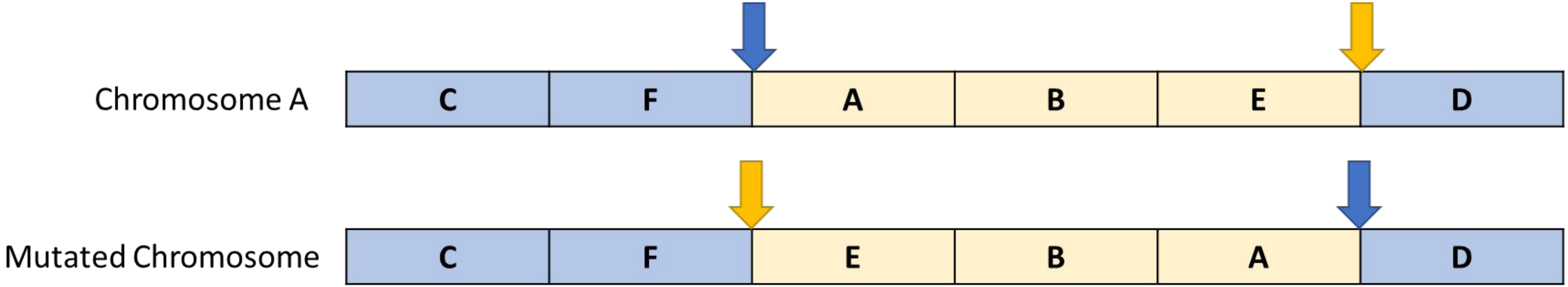
Implementations

- Adapting GABONST for TSP optimization
  - Two Point Crossover Implementation



Implementations

- Adapting GABONST for TSP optimization
  - 2-Opt Mutation Implementation



Evaluations

- Setup to optimize n-dimensional real-valued problems
- Experiment 1
  - 15 objective functions (shown in table 1)
  - Monte Carlo simulation of 50 runs
  - 100 iterations with population size of 50 for each run
- Experiment 2
  - TSP Optimization with GABONST

Experiment 1

Table 1. Details of the utilized mathematical objective functions.

Objective Function	Dim	Range	Optimal Solution
$f_1(x) = -\frac{1}{d}\sum_{i=1}^{d}\sin^6(5\pi x_i)$	10	[−1, 1]	−1
$f_2(x) = -\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	2	[0, π]	−1.8013
$f_3(x) = (x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	2	[−10, 10]	0
$f_4(x) = \frac{1}{2}\sum_{i=1}^{d}\left(x_i^4-16x_i^2+5x_i\right)$	10	[−5, 5]	−391.6599
$f_5(x) = \prod_{i=1}^{d}\sqrt{x_i}\sin(x_i)$	2	[0, 10]	−6.1295
$f_6(x) = \sum_{i=1}^{n}x_i^2$	256	[−5.12, 5.12]	0
$f_7(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	30	[−30, 30]	0
$f_8(x) = \sum_{i=1}^{n}ix_i^4+\mathrm{random}(0,1)$	30	[−1.28, 1.28]	0
$f_9(x) = \sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	[−5.12, 5.12]	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	128	[−32.768, 32.768]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[−600, 600]	0
$f_{12}(x) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[−2, 2]	3
$f_{13}(x) = \sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[−5, 5]	0.00030
$f_{14}(x) = 4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5, 5]	−1.0316
$f_{15}(x) = \left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	[−5, 5]	0.398

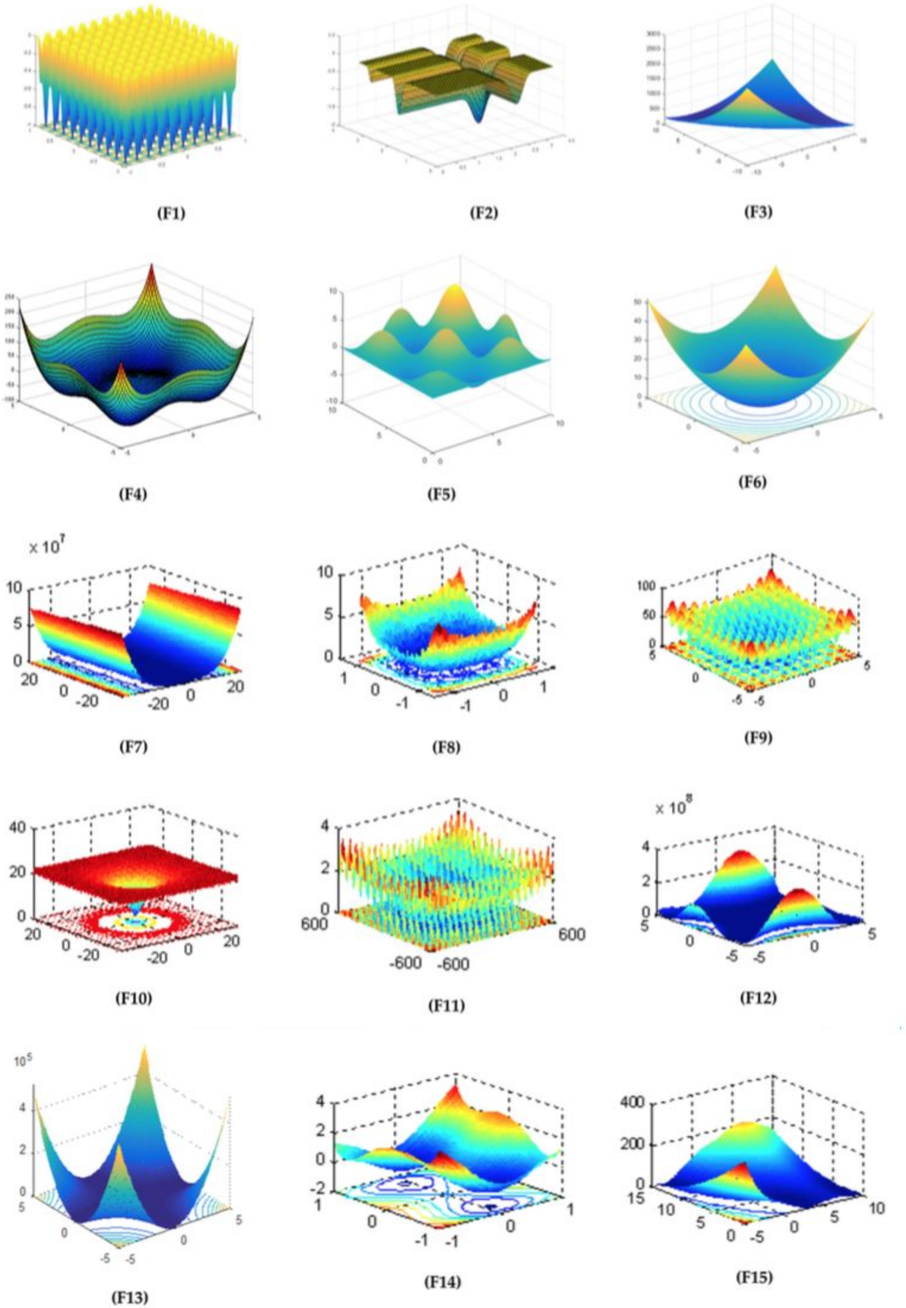


Figure 4. Graphical representation of mathematical objective functions (mathematical objective functions **F1–F15** in Table 1).

Our Experiment 1 Results

Max Iteration = 100, Population Size = 50, Mutation Rate = 0.01, Gamma = 0.4

Statistics of 50 Simulations

Function	Dims	Optima	Paper-Mean	Our-Mean	Paper-Std	Our-Std	Paper-RMSE	Our-RMSE
f_1	10	-1.0	-0.99688	-0.765486	0.008434	0.0315088	0.008912	1.67287
f_2	2	-1.8013	-1.9511	-1.84056	2.24299e-15	0.000328401	0.1498	0.277628
f_3	2	0.0	0.0	0.0272772	0.0	0.0275747	0.0	0.272874
f_4	10	-391.66	-380.897	-323.345	11.2675	13.1511	15.502	491.755
f_5	2	-6.1295	-6.1295	-6.1252	4.48598e-15	0.00436423	0.0	0.0431136
f_6	256	0.0	0.0	1090.45	0.0	45.7872	0.0	7717.32
f_7	30	0.0	0.0	2.11213e7	0.0	7.18066e6	0.0	1.57581e8
f_8	30	0.0	0.00015895	11.4158	0.00015852	2.71969	0.00022336	82.9367
f_9	30	0.0	0.0	285.247	0.0	15.7535	0.0	2020.01
f_10	128	0.0	0.0	19.7894	0.0	0.156269	0.0	139.937
f_11	30	0.0	0.0	136.766	0.0	23.0032	0.0	980.395
f_12	2	3.0	2.99999	3.18065	1.8266e-15	0.201796	7.6591e-14	1.90448
f_14	2	-1.0316	-1.03163	-1.02354	5.4942e-16	0.0085436	2.8453e-5	0.0826142
f_15	2	0.398	0.39788	0.401561	3.3645e-16	0.00429574	0.0001126	0.0392199

TSP Benchmark on 50 Generated Cities

```
50×50 Matrix{Float64}:
 0.0         0.358745   0.594028   0.123579   …  0.710117   0.282903   0.54627
 0.358745    0.0        0.57837    0.456627      0.778443   0.372911   0.599314
 0.594028    0.57837    0.0        0.706758      0.225779   0.311127   0.111998
 0.123579    0.456627   0.706758   0.0           0.804043   0.39725    0.649534
 0.387598    0.46363    0.214319   0.495452      0.335897   0.111767   0.160895
 0.318055    0.0887968  0.639932   0.399473   …  0.828748   0.400419   0.648385
 0.498317    0.642847   0.262829   0.581829      0.230849   0.273318   0.151205
 0.212139    0.149763   0.581065   0.306958      0.752804   0.310762   0.573376
 0.389834    0.429837   0.926496   0.359482      1.0796     0.632974   0.905042
 0.661618    0.647309   0.0713539  0.772374      0.186043   0.379011   0.143315
 0.539474    0.19298    0.577174   0.645158   …  0.798664   0.472837   0.632046
 0.179845    0.534056   0.633737   0.157383      0.693016   0.346408   0.557588
 0.942702    0.877672   0.34905    1.05576       0.349946   0.659911   0.42147
 ⋮                                            ⋱
 0.734031    0.704734   0.140747   0.845332      0.201047   0.45122    0.212168
 0.617207    0.839647   0.47726    0.664158      0.346606   0.468506   0.367312
 0.393566    0.46269    0.206615   0.502238   …  0.33298    0.115739   0.156562
 0.391017    0.685708   0.539188   0.412233      0.516132   0.36075    0.435436
 0.622013    0.590136   0.034106   0.736109      0.236252   0.33941    0.143831
 0.475021    0.62855    0.275475   0.557344      0.255572   0.257043   0.165361
 0.621625    0.783097   0.345194   0.692716      0.202389   0.413223   0.241335
 0.766429    0.509497   0.450565   0.888575   …  0.661688   0.562861   0.552802
 0.277266    0.118245   0.486887   0.390943      0.674686   0.255711   0.494382
 0.710117    0.778443   0.225779   0.804043      0.0        0.447375   0.180381
 0.282903    0.372911   0.311127   0.39725       0.447375   0.0        0.272296
 0.54627     0.599314   0.111998   0.649534      0.180381   0.272296   0.0
```

TSP Benchmark on 50 Generated Cities

Max Iteration = 30000, Population Size = 200, 2-Opt Mutation Rate = 1%

Simulations = 20

Initial Population Score	Best Scores	Initial NN Population Score	Best Scores NN
26.3026	5.9795	26.1263	5.9654
26.2593	5.80073	26.0245	6.23263
26.1648	6.18643	25.905	5.7934
26.2737	6.53165	26.0557	5.85588
26.207	5.86557	25.99	6.09551
26.267	6.09848	25.9621	6.12151
26.3573	6.43266	26.0609	5.85753
26.1914	6.06831	25.985	5.77073
26.2095	5.88087	25.9066	5.83225
26.1871	6.00937	25.9876	6.16843
26.4239	6.10426	26.2258	6.3293
26.17	5.83535	25.9839	6.01774
26.2707	6.05152	26.0002	6.58154
26.2911	5.89869	26.0631	5.9611
26.2432	6.0221	25.9947	6.05207
26.1798	5.75201	25.9058	6.14055
26.233	5.83374	26.0108	5.94477
26.2133	5.82825	25.9104	6.04264
26.2567	6.29098	26.0301	5.94644
26.2914	5.80793	26.0518	6.14048

TSP Benchmark on 50 Generated Cities

Max Iteration = 50000, Population Size = 200, 2-Opt Mutation Rate = 1%

Simulations = 20

Initial Population Score	Best Scores	Initial NN Population Score	Best Scores NN
26.3026	5.9795	26.1263	5.71878
26.3105	5.70858	26.0758	5.74499
26.147	5.80793	25.9791	6.1561
26.4116	5.7299	26.0636	5.86989
26.3412	5.93142	26.1128	5.72389
26.1436	6.12415	25.8443	5.796
25.9356	5.74877	25.7007	5.81219
26.2118	5.95429	25.9631	6.12604
26.0168	6.11278	25.7992	5.95472
26.0508	5.81284	25.8834	6.01208
26.375	6.19297	26.0865	5.81731
26.3068	6.02814	26.0218	5.92182
26.1074	5.71878	25.9662	5.79931
26.1507	5.80924	25.8864	5.92148
26.2042	5.84586	26.0276	6.05866
26.2297	5.95001	25.9855	5.79172
26.3514	6.01489	26.1316	5.71878
26.2843	5.83286	26.0828	5.74863
26.1633	5.98994	25.9697	5.7452
26.353	5.95023	26.1431	5.98274

TSP Benchmark on TSPlib Benchmark

Max Iteration = 10000, Population Size = 200, 2-Opt Mutation Rate = 1%
Simulations = 10

TSP	Cities	Optimal	GABONST - Best Score	Differ From Optimal
Berlin52	52	7542.0	9188.2	+21.8%
EIL51	51	426.0	518.7	+21.6%
CH130	130	6110.0	18348.7	+200.3%
LIN105	105	14379.0	39471.5	+174.5%
KROA100	100	21282.0	56031.6	+163.3%

TSP Benchmark on TSPlib Benchmark

Max Iteration = 10000, Population Size = 200, 2-Opt Mutation Rate = 1%
Simulations = 10, with Nearest Neighbor Initializations

TSP	Cities	Optimal	GABONST - Best Score	Differ From Optimal
Berlin52	52	7542.0	7791.8	+3.3%
EIL51	51	426.0	451.2	+5.9%
CH130	130	6110.0	6933.7	+13.5%
LIN105	105	14379.0	16685.8	+16.0%
KROA100	100	21282.0	24535.6	+15.3%

TSP Art

570 Cities generated by dithering the grey image,
population = 600, Iterations = 50000
Initialized with 570 nearest neighbor populations

Discussions

- Biodiversity (given enough time it can find the solution)
    - Path “D” introduces random search to the population
- Discrepancies in objective function benchmark (no response from author)
    - Our implementation significantly underperforms with higher dimension
- Lack of elitism

Another Implementation In Python [2]

We found a repository of an GABONST implementation in Python as we tried to verify our implementation.

In their implementation, instead of mutating with mutation rate, they force 1 gene in the chromosomes to be mutated in each mutation operation.

The result in 14/15 objective functions in Experiment 1 matches better with our implementation (especially the high dimension functions) versus the published results in paper

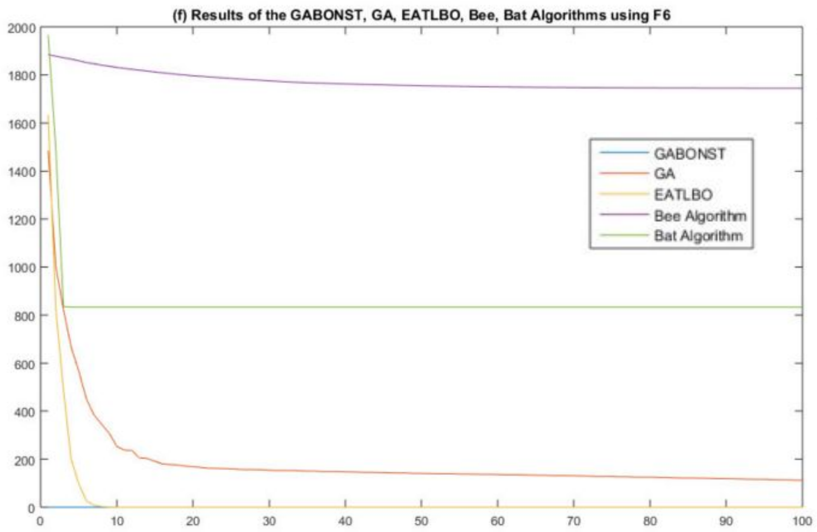
[2] Source: https://colab.research.google.com/drive/12Y5d6MB6bgpXX9XOaon91gVQ3YBOH2S8?usp=sharing

Comparison Between Implementations

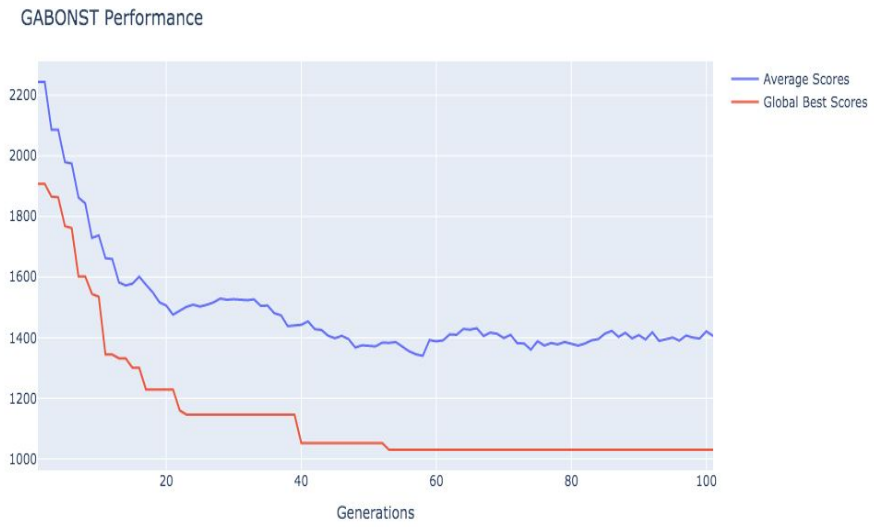
Single Run on 256-dimension Sphere Function

Search Space: [-5.12, 5.12], Optimal = 0

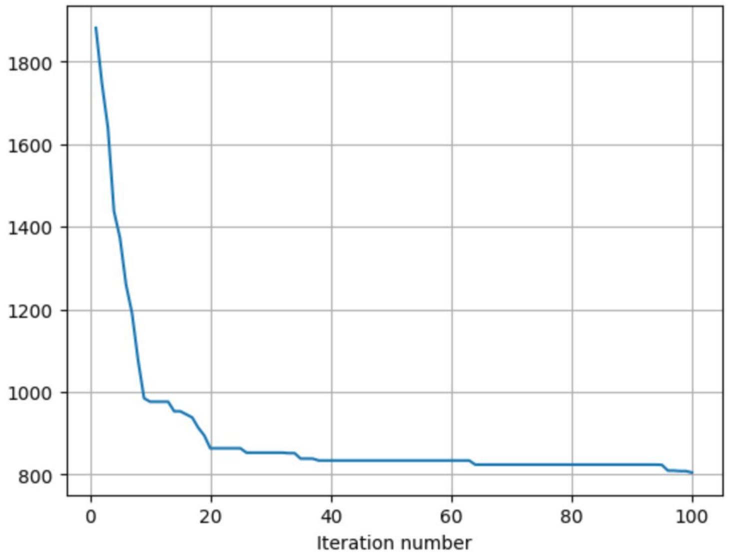
Max Iteration = 100, Population Size = 50, Mutation Rate = 1%*, Gamma = 0.4



Template Paper
Result: 0 (Optimal)



Our Implementation
Result: 1029.819



Python Implementation [2]
Result: 805.024

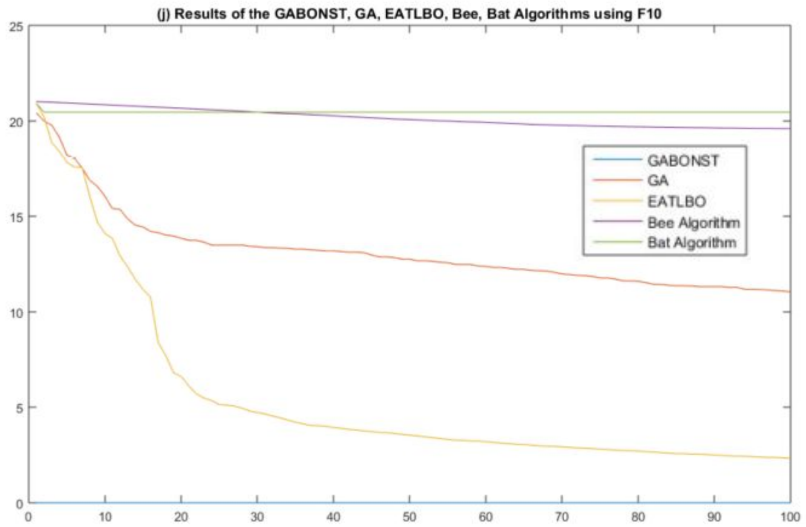
*Python implementation forces one and only one chromosome to be mutated in every mutation operation

Comparison Between Implementations

Single Run on 128 dimensional function F10

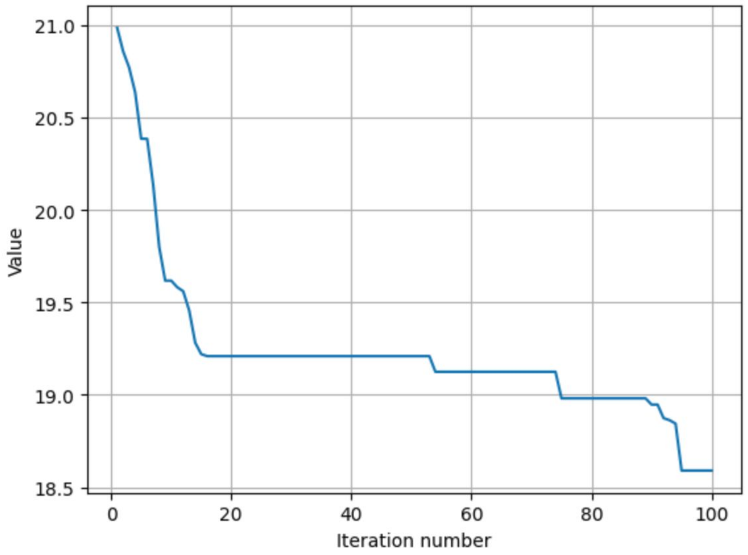
Search Space: [-32.768, 32.768], Optimal = 0

Max Iteration = 100, Population Size = 50, Mutation Rate = 1%*, Gamma = 0.4



Template Paper
Result: 0 (Optimal)

Our Implementation
Result: 19.869



Python Implementation [2]
Result: 18.59

*Python implementation forces one and only one chromosome to be mutated in every mutation operation

Reference

1. Albadr, Musatafa Abbas Abbood et al. "Genetic Algorithm Based on Natural Selection Theory for Optimization Problems." *Symmetry* 12 (2020): 1758.

2. https://colab.research.google.com/drive/12Y5d6MB6bgpXX9XOaon91gVQ3YBOH2S8?usp=sharing

Thank You

Backup Slides

Results of Experiment 1

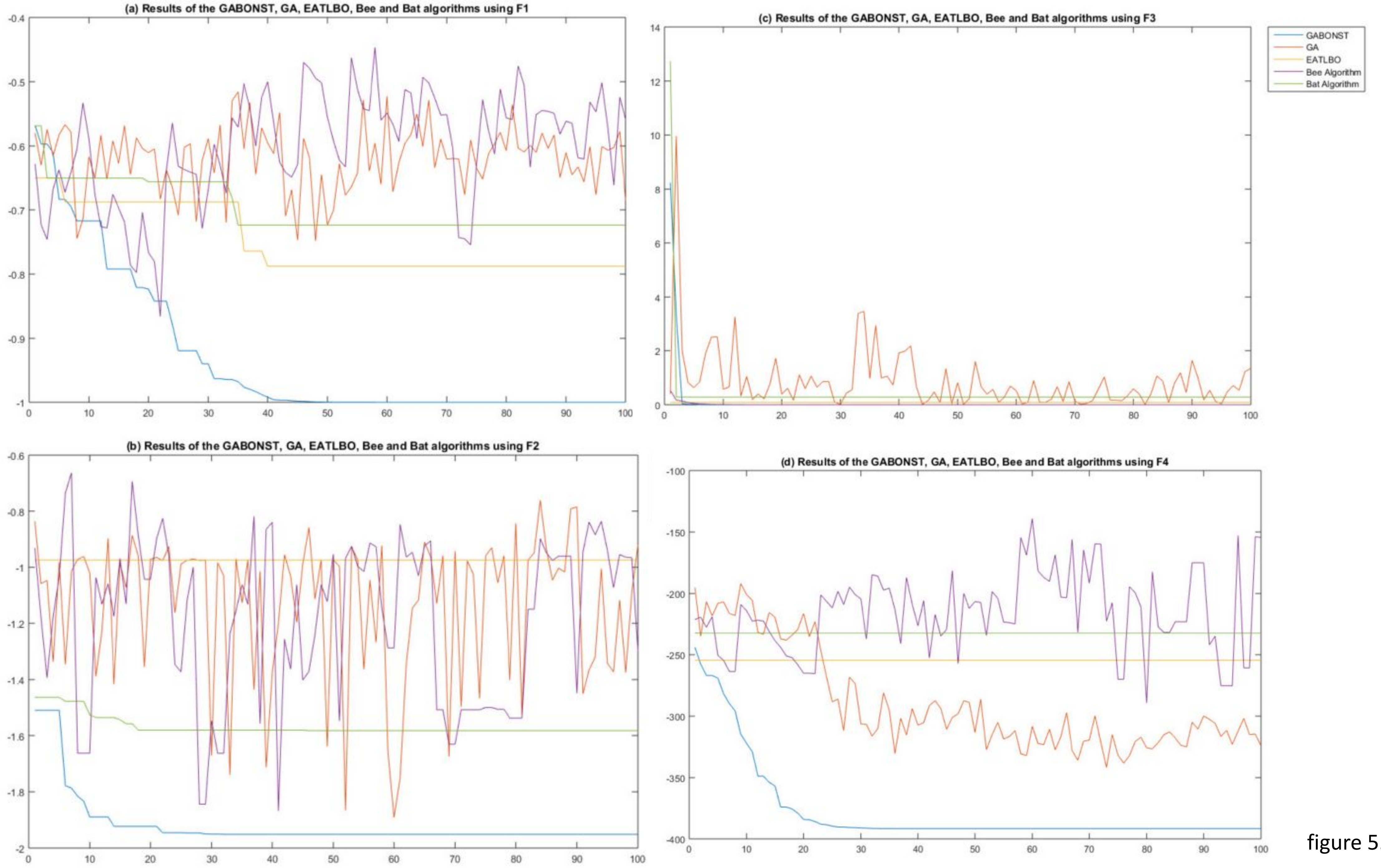
- The following statistics from experiment 1 are captured in table 2
  - Root mean square error (RMSE)
  - Mean cost
  - Standard deviation (STD)
- GABONST outperforms the competing algorithms in most cases
  - Except GA being slightly better on F14
  - GABONST and EATLBO both reached optimal on F11 (tie)
- GABONST exhibit fast convergence as shown in figure 5

Table 2. Statistical results of the mathematical objective functions for the optimization approaches (GABONST, GA, enhanced ameliorated teaching learning-based optimization (EATLBO), Bat and Bee)).

	F1	F2	F3	F4	F5
GA–RMSE	0.408266	0.66005	0.457172	69.84426	2.734866
GABONST–RMSE	**0.008912**	**0.1498**	**0**	**15.50201**	**0**
EATLBO–RMSE	0.241362	0.803974	0.379706	148.6135	0.715572
Bat-RMSE	0.3820	0.4033	5.7316	153.3258	0.2902
Bee-RMSE	1.0000	0.5693	9.2615×10^{-10}	186.7217	0.2444
GA–Mean	−0.60383	−1.20717	18.4889	−326.997	−3.7482
GABONST–Mean	**−0.99688**	**−1.9511**	**0**	**−380.897**	**−6.1295**
EATLBO–Mean	−0.77554	−0.99733	0.27178	−245.971	−5.67757
Bat-Mean	−0.6215	−1.5889	3.3044	−240.5929	−5.9602
Bee-Mean	-9.4481×10^{-11}	−1.3024	6.1739×10^{-10}	−207.6444	−6.0014
GA–STD	0.099654	0.290461	0.271562	26.66188	1.358616
GABONST–STD	0.008434	**2.24299×10^{-15}**	**0**	**11.26751**	**4.48598×10^{-15}**
EATLBO–STD	0.089637	0.00257	0.267856	29.62479	0.56043
Bat-STD	0.0526	0.3463	4.7307	26.4876	0.2381
Bee-STD	**3.1940×10^{-11}**	0.2769	6.9736×10^{-10}	31.9960	0.2103
	F6	**F7**	**F8**	**F9**	**F10**
GA–RMSE	115.0308	8.1381×10^{3}	0.1612	42.7116	10.9405
GABONST–RMSE	**0**	**0**	**2.2336×10^{-4}**	**0**	**0**
EATLBO–RMSE	4.4872×10^{-56}	28.9495	**2.7609×10^{-4}**	205.0804	2.8037
Bat-RMSE	1.2039×10^{3}	9.0087×10^{7}	74.3799	364.5571	20.1364
Bee-RMSE	1.7538×10^{3}	2.3143×10^{7}	14.6593	141.4336	19.6219
GA–Mean	113.9390	6.7387×10^{3}	0.1464	41.6742	10.9243
GABONST–Mean	**0**	**0**	**1.5895×10^{-4}**	**0**	**0**

Table 2. *Cont.*

EATLBO–Mean	3.3624×10^{-56}	28.9495	2.0063×10^{-4}	202.5195	2.6162
Bat-Mean	1.1761×10^{3}	8.1520×10^{7}	67.4795	362.3866	20.1321
Bee-Mean	1.7533×10^{3}	2.1898×10^{7}	14.3940	140.4960	19.6218
GA–STD	15.9710	4.6090×10^{3}	0.0682	9.4516	39.3751
GABONST–STD	**0**	**0**	$\mathbf{1.5852 \times 10^{-4}}$	**0**	**0**
EATLBO–STD	3.0016×10^{-56}	0.0175	1.9160×10^{-4}	32.6362	1.0182
Bat-STD	259.8600	3.8731×10^{7}	31.6047	40.1246	0.4206
Bee-STD	39.3751	7.5653×10^{6}	2.8049	16.4241	0.0496
	F11	**F12**	**F13**	**F14**	**F15**
GA–RMSE	2.2342	6.4588×10^{-6}	0.0018	$\mathbf{2.8284 \times 10^{-5}}$	1.1264×10^{-4}
GABONST–RMSE	**0**	$\mathbf{7.6591 \times 10^{-14}}$	$\mathbf{1.4195 \times 10^{-4}}$	2.8453×10^{-5}	$\mathbf{1.1264 \times 10^{-4}}$
EATLBO–RMSE	**0**	9.9516	0.0067	0.0372	0.1170
Bat-RMSE	348.4865	21.6007	0.0645	0.8243	0.5704
Bee-RMSE	221.4966	2.5635×10^{-9}	2.7909×10^{-4}	2.8453×10^{-5}	1.1264×10^{-4}
GA–Mean	2.1932	3.000000925712561	0.0017	**−1.031628252987515**	0.3978873583048
GABONST–Mean	**0**	**2.999999999999923**	$\mathbf{4.0025 \times 10^{-4}}$	−1.031628453489878	**0.3978873577297**
EATLBO–Mean	**0**	9.9257	0.0042	−1.0078	0.4496
Bat-Mean	338.7847	18.6428	0.0464	−0.4807	0.7070
Bee-Mean	219.4071	3.000000001676081	5.1854×10^{-4}	−1.031628453353341	0.3979
GA–STD	0.4302	6.4570×10^{-6}	0.0011	1.3312×10^{-6}	2.3933×10^{-9}
GABONST–STD	**0**	$\mathbf{1.8266 \times 10^{-15}}$	$\mathbf{1.0152 \times 10^{-4}}$	$\mathbf{5.4942 \times 10^{-16}}$	$\mathbf{3.3645 \times 10^{-16}}$
EATLBO–STD	**0**	7.2188	0.0055	0.0288	0.1061
Bat-STD	82.4855	15.0473	0.0455	0.6194	0.4843
Bee-STD	30.6602	1.9593×10^{-9}	1.7534×10^{-4}	2.0841×10^{-10}	1.0905×10^{-10}



(a) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F1
(b) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F2
(c) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F3
(d) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F4
GABONST
GA
EATLBO
Bee Algorithm
Bat Algorithm

figure 5.

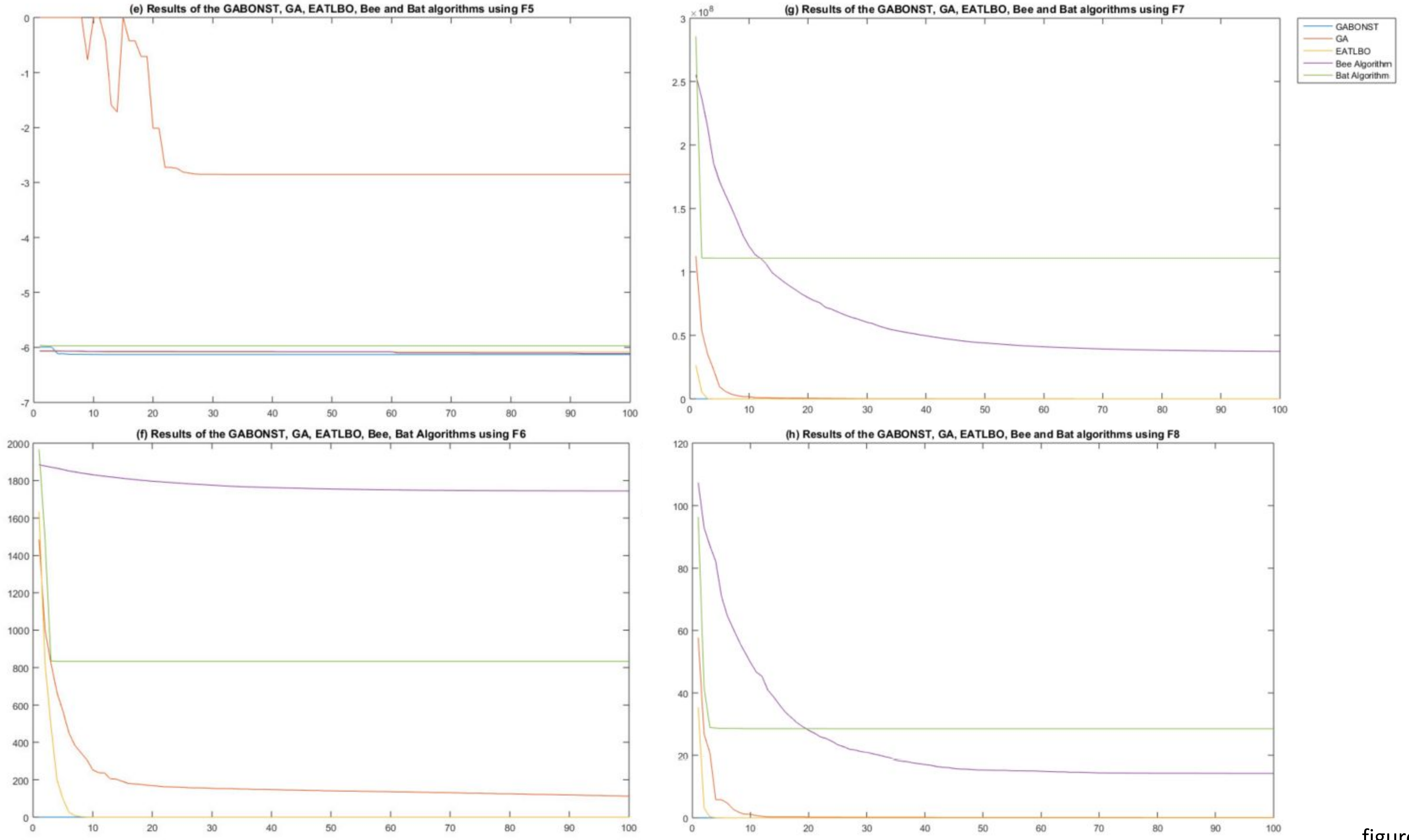

(e) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F5
(g) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F7
$\times 10^8$
GABONST
GA
EATLBO
Bee Algorithm
Bat Algorithm
(f) Results of the GABONST, GA, EATLBO, Bee, Bat Algorithms using F6
(h) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F8

figure 5.

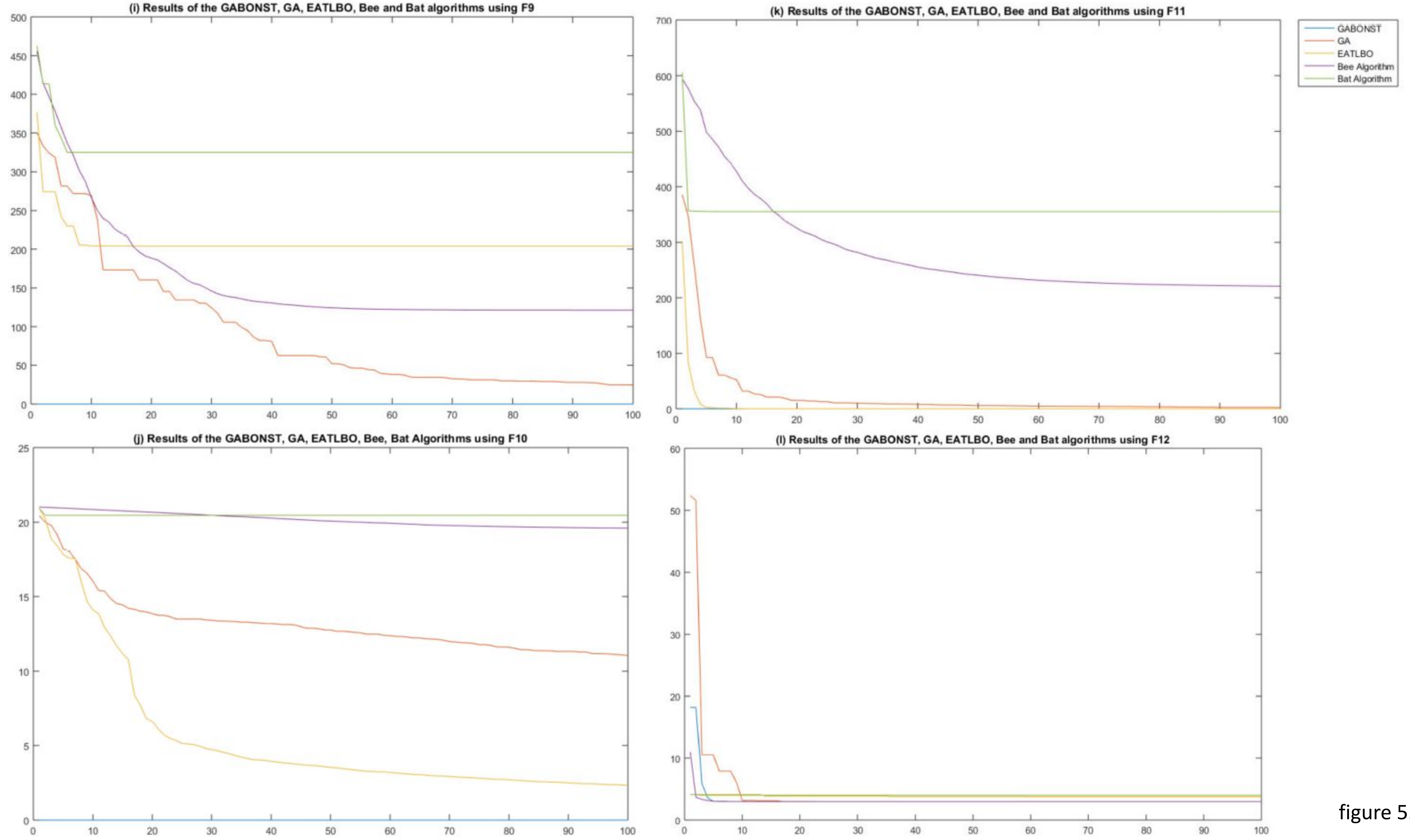

(i) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F9
(k) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F11
GABONST
GA
EATLBO
Bee Algorithm
Bat Algorithm
(j) Results of the GABONST, GA, EATLBO, Bee, Bat Algorithms using F10
(l) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F12

figure 5.

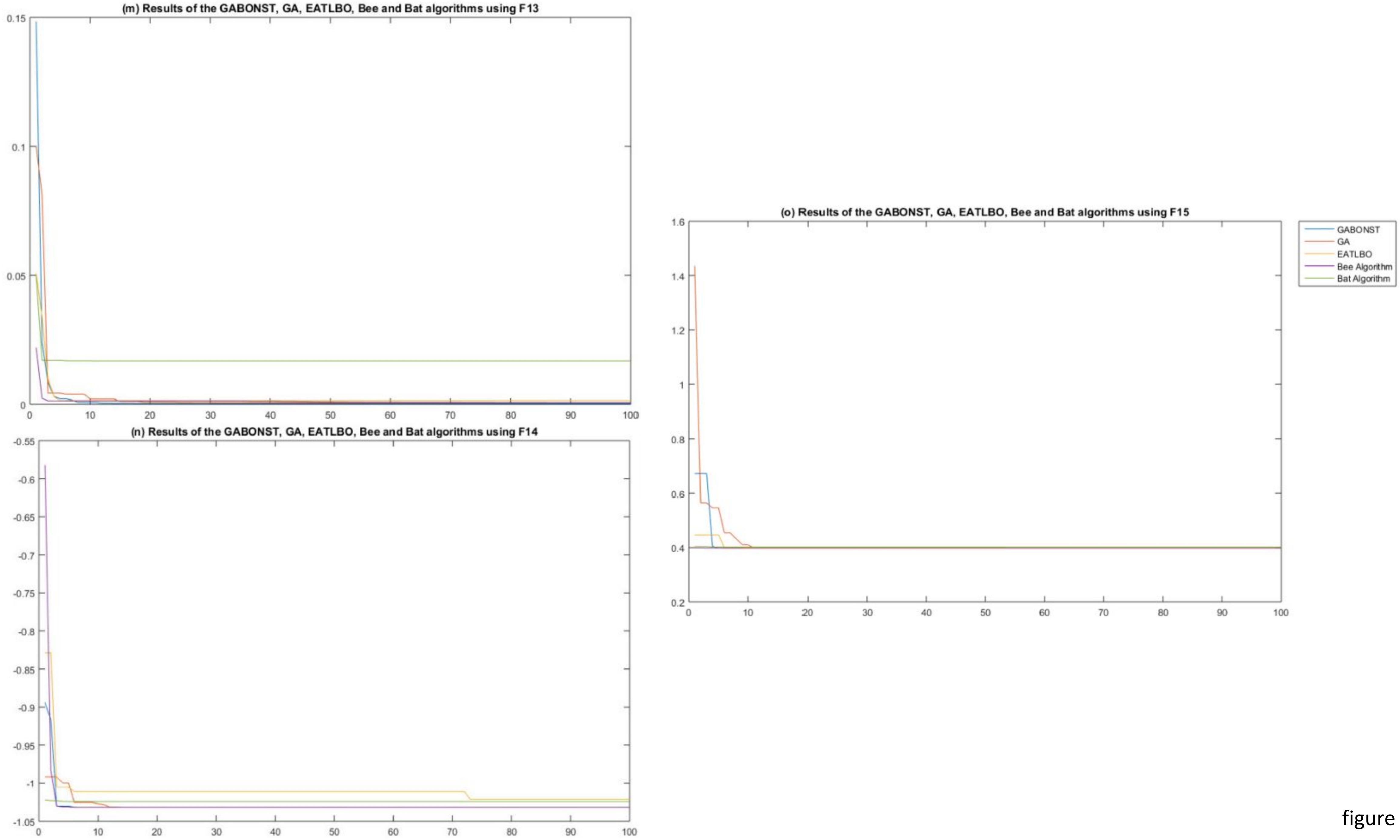

(m) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F13
(n) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F14
(o) Results of the GABONST, GA, EATLBO, Bee and Bat algorithms using F15
GABONST
GA
EATLBO
Bee Algorithm
Bat Algorithm

figure 5.