Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 9541-9562 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

A STUDY OF THE COMBINED USE OF DIFFERENTIAL EVOLUTION AND GENETIC ALGORITHMS

Eduardo K. da Silva and Helio J. C. Barbosa

Laboratório Nacional de Computação Científica – LNCC,
Petrópolis – RJ, Brasil, {krempser, hcbm}@lncc.br

Keywords: Metaheuristics, Differential Evolution, Genetic Algorithms, Hybrid.

Abstract. Differential Evolution (DE) and Genetic Algorithms (GA) are efficient stochastic, population-based, metaheuristics for global optimization, that are widely used in many different fields. In order to explore the qualities of each algorithm at different times of the search, we used the algorithms in an interleaved way, evaluating them on different constrained optimization test problems from the literature. With this it was possible to observe the behavior of the final algorithm, a hybrid one, and the extent to which it alters the moment and number of times each component algorithm is used. Furthermore it was possible to perform comparisons between the proposed hybrid and the models usually adopted for the original algorithms.

1 INTRODUCTION

The metaheuristics are used widely in many fields, and among in the metaheuristics we can highlight the called Genetic Algorithms (GA), and more recently, the Differential Evolution (DE).

The emphasis on these algorithms is due to the high-efficiency obtained by them in several areas. However, each algorithm has its specifications, traversing different paths in the search space.

For such specifications, each algorithm is best suited to different problems and to make choice of the algorithm is a new optimization problem. Against this background emerge the hybrid algorithms, which combine the known techniques.

The usefulness of hybrid algorithms is demonstrated when applied in constrained problems where there is large differences in the results obtained by the same algorithm on different problems. We can also think that each algorithm could be more effective when applied at certain times of the search, because the search features (exploration and exploitation, for example) are more appropriate at different times of the optimization process.

The factors mentioned lead us to develop an algorithm that makes full use of DE and GA in different stages of the search - different generations. The way in which we decide when and how long each method will be used is set adaptive, considering the solutions obtained during the optimization process.

In the next section we present the optimization problem considered here, the Differential Evolution and Genetic Algorithms are briefly described in sections 3 and 4. The proposed method is presented in section 5 and the computational experiments shown in section 6. The conclusions are presented in section 7.

2 THE OPTIMIZATION PROBLEM

The class of constrained optimization problems considered here can be written as

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_j(x) \geq 0, \quad j = 1, \ldots, p \\ & h_k(x) = 0, \quad k = 1, \ldots, q \\ & x^L \leq x \leq x^U \end{array} \tag{1}$$

where $f(x)$ is the objective function to be minimized, $x \in R^n$ is the vector of design variables with lower and upper bounds defined by x^L and x^U, respectively, and p and q are the number of inequality and equality constraints, respectively.

It is important to note that the constraints h and g are complex *implicit* functions of the design variables and no differentiability is assumed for f, g, and h.

3 DIFFERENTIAL EVOLUTION

The original proposal of DE by Storm and Price (Storn and Price, 1997) presents a simple and efficient algorithm for global optimization over continuous spaces.

Algorithm (1), from (Mezura-Montes et al., 2006b), shows the pseudo-code for the DE/rand/1/bin variant. In this variant the individuals involved in the mutation process are selected randomly.

The main variants of the DE modify the way that the individuals are selected for participate of the mutation, which in the original proposal were randomly (called DE/rand/1/bin), it's basically change the line 11 of the algorithm 1, example are:

Algorithm 1 Algorithm DE/rand/1/bin

```
1:  procedure DE(POP, GEN, F, CR)
2:      G = 0
3:      CreateRandomInitialPopulation(POP)
4:      Evaluate f(x⃗_{i,G})                                   ▷ ∀i, i = 1, ..., POP
5:      for G = 1 : GEN do
6:          for i = 1 : POP do
7:              SelectRandomly(r1, r2, r3)                    ▷ r1 ≠ r2 ≠ r3
8:              jRand = randInt(1, D)                         ▷ D is the dimension of the problem
9:              for j = 1 : D do
10:                 if (rand(0, 1) < CR or j = jRand) then
11:                     u_{i,j,G+1} = x_{r3,i,G} + F(x_{r1,j,G} − x_{r2,j,G})
12:                 else
13:                     u_{i,j,G+1} = x_{i,j,G}
14:                 end if
15:             end for
16:             if f(u_{i,G+1}) ≤ f(x_{i,G}) then
17:                 x⃗_{i,G+1} = u⃗_{i,G+1}
18:             else
19:                 x⃗_{i,G+1} = x⃗_{i,G}
20:             end if
21:         end for
22:     end for
23: end procedure
```

DE/best/1/bin Proposed in (Price, 1999) modify the initial model using in the mutation the best individual in the population with the base vector. The mutation is described by: $u_{j,i} = x_{j,best} + F_j(x_{j,r_1} - x_{j,r_2})$, with $x_{j,best}$ the best individual in the population in the current generation and r_1 e r_2 randomly selected individuals

DE/target-to-best/1/bin This variant, described in (Price, 1999), proposes the joint use of the best individual of the population and the target individual (individual that will be used in the comparison after the mutation). So, these two information are combined in the mutation operator: $u_{j,i} = x_{j,i} + F_j(x_{j,best} - x_{j,i}) + F(x_{j,r_1} - x_{j,r_2})$

Another variants can be found in (Mezura-Montes et al., 2006a; Qing, 2008), techniques that combine several variants also were developed, such as in (Silva et al., 2010). In this paper we used the original proposal (DE/rand/1/bin).

4 GENETIC ALGORITHMS

Genetic Algorithms (GA) are stochastic, population nature inspired search procedures which have found applications in different areas and have been shown efficiently search complex spaces for good solutions to optimization problems.

These algorithms are one of the most traditional and widely used evolutionary algorithms.

The GA encodes all the variables x_i corresponding to a candidate solution in a chromosome and maintain a population of candidate solutions which is evolved mimicking Natures's evolutionary process: solutions are selected by a stochastic process that favors better solutions and

have their genetic material recombinated/mutated by means of genetic operators. This gives rise to a new population with improved solutions. The process starts from a usually random initial population and is repeated for a given number of generations or until some stopping criteria are met.

There are several schemes of selection to the application of the genetic operators as well as there are a great number of genetic operators already developed. Here a rank-based selection scheme (Whitley, 1998) was used and the genetic operators adopted are listed below:

- The one-point (1X) crossover operator is the analogue to the standard one-point crossover for binary coded GAs. Is generates two offspring by exchanging the alleles after the randomly chosen position int the parent's chromosomes.

- Deb and Agrawal (Deb and Agrawal, 1994) developed the Simulated Binary Crossover (SBX), which simulates the working principle of the single-point crossover operator on binary strings in continuous domain.

- Eshelman and Schaffer (Eshelman and Schaffer, 1993) introduced the concept of interval schemata for real-coded GAs and suggested a blend crossover (BLX-α) operator for two parents. (BLX-α).

- The random mutation (RM) simply generates an offspring by setting a randomly selected allele of the parent chromosome to a randomly chosen value uniformly distributed over the range of such allele $[x^L, x^U]$.

- Another mutation operator applied performed an increment Δ (DM) to each variable with probability (p_m) in (0,1) to be applied

$$x = p + \delta\Delta_{max}$$

  where p is the parent, x is the offspring and Δ_{max} is a fixed quantity, it represent the maximum permissible change in the parent. The δ is a random number.

- The non-uniform mutation (NUM) operator (Michalewicz, 1992), when applied to an individual x_i at generation *gen* and when the total number of generations allowed is *maxgen*, mutates a randomly chosen variable x_i according to

$$x_i \leftarrow \begin{cases} x_i + \Delta(gen, x^U - x_i) & \text{if} \quad \tau = 0 \\ x_i - \Delta(gen, x_i - x^L) & \text{if} \quad \tau = 1 \end{cases}$$

  where x^L and x^U are respectively the lower and upper bounds for the variables x_i, τ is randomly chosen as 0 or 1 and the function $\Delta(gen, y)$ is defined as

$$\Delta(gen, y) = y(1 - \mu^{(1-\frac{gen}{maxgen})^\eta})$$

  with μ randomly chosen in [0, 1] and the parameter η set to 2.

In this paper the recombination operators were used with probabilities: 1%(1X), 60%(SBX) e 30%(BLX-α), as well as for mutation operators: 30%(RM), 40%(DM) and 20%(NUM).

5 THE HYBRID ALGORITHM

This paper presents a hybrid and adaptive algorithm that combines GA and DE. Allowing it to adjust the use of such algorithms during the search process. The adaptive technique considered here (Barbosa and Sá, 2000) is based on collecting information about the performance or productivity of each method.

We used the algorithms in an interleaved way, in every generation an algorithm is chosen following its probabilities. The probabilities are adjusted for productivity, and a method is said productive if produces an individual better than all individuals of the previous population – a new best.

In order to accumulate the reward, we follow Lobo and Goldberg (Lobo and Goldberg, 1996) and compute the reward R_i of the i-th method through the expression

$$R_i = (1 - c)R_i + cw_i \tag{2}$$

where w_i is the instantaneous reward and c is a positive parameter which controls the amount of memory in the process. Larger values of c correspond to less memory in the sense that the importance of the present reward is increased as compared to performance in the past.

The probability of each method is then redefined by the expression

$$p_i = \frac{R_i}{\sum_1^n R_i} \tag{3}$$

The instantaneous reward w_i is calculated for each generation if the new population produced a new best individual. The difference of the fitness of the new best individual and previous best determines the reward for the algorithm of the current generation.

For constrained problems the reward is obtained similarly to the method of handling constraints used, a binary tournament described by (Deb, 2000). In other words, if both individuals are feasible the reward is the difference of the fitness, if one is not feasible then the reward is the sum of the constrained of the individual not feasible.

6 COMPUTATIONAL EXPERIMENTS

In this section we present the results of some computational experiments obtained when we applied the proposed algorithm to the G–24 set problem (Liang et al., 2006), which is extensively used as benchmark. We tested several values for the parameter c and evaluation the behavior of the probabilities of each algorithm during the process.

For all problems we used forty individuals in the population and 5000 generations were allowed. Due to the limit on the number of evaluations only in fifteen problems some algorithm found a valid solution. The results for these problems are present below.

Details of the 24 test problems are presented in the Table 1, where n is the number of decision variables, LI is the number of linear inequality constraints, NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints.

The Tables 2, 3, 4, 5 and 6 present the results for each algorithm and the results found with different values of c. Moreover, the results for a fixed set of probabilities for DE and GA are displayed. These probabilities were chosen based in the best results of the adaptive scheme used for each problem, the probabilities correspond the number of times that each algorithm was used by the total number of generations.

Problem	n	Type	LI	NI	LE	NE
G–01	13	quadratic	9	0	0	0
G–02	20	nonlinear	0	2	0	0
G–03	10	polynomial	0	0	0	1
G–04	5	quadratic	0	6	0	0
G–05	4	cubic	2	0	0	3
G–06	2	cubic	0	2	0	0
G–07	10	quadratic	3	5	0	0
G–08	2	nonlinear	0	2	0	0
G–09	7	polynomial	0	4	0	0
G–10	8	linear	3	3	0	0
G–11	2	quadratic	0	0	0	1
G–12	3	quadratic	0	1	0	0
G–13	5	nonlinear	0	0	0	3
G–14	10	nonlinear	0	0	3	0
G–15	3	quadratic	0	0	1	1
G–16	5	nonlinear	4	34	0	0
G–17	6	nonlinear	0	0	0	4
G–18	9	quadratic	0	13	0	0
G–19	15	nonlinear	0	5	0	0
G–20	24	linear	0	6	2	12
G–21	7	linear	0	1	0	5
G–22	22	linear	0	1	8	11
G–23	9	linear	0	2	3	1
G–24	2	linear	0	2	0	0

Table 1: Details of the 24 test problems

Where **Algorithm** is the algorithm or configuration applied, **Best**, **Avg**, **Worst**, **Std** are statistical results after twenty runs, **FR** is the number of runs which the algorithm found feasible individuals. The values **pDE** and **pGA** indicate the correlation between the number of times that each algorithm was used by the total number of generations.

The average of the probabilities of each algorithm during the search are shown in the Fig. 1 to 8.

We can observe different variations of the probabilities for each c value and, as expected, larger values of c correspond to less memory and with this there is greater freedom to change, but if we discard the past can devalue a method very quickly and so generate worse results.

Another important result shown by graphs are the changes of behavior during process,which indicates the adjust of the algorithm in every moment of the search.

The pDE and pGA values show the percentage of use of each technique (DE and GA). Thus, we can see that in most cases the technique with better results were also the most used by hybrid algorithm. Moreover, the results of the function G–09 shown that while the techniques have not obtained good results separately, the hybrid algorithm obtained improvements.

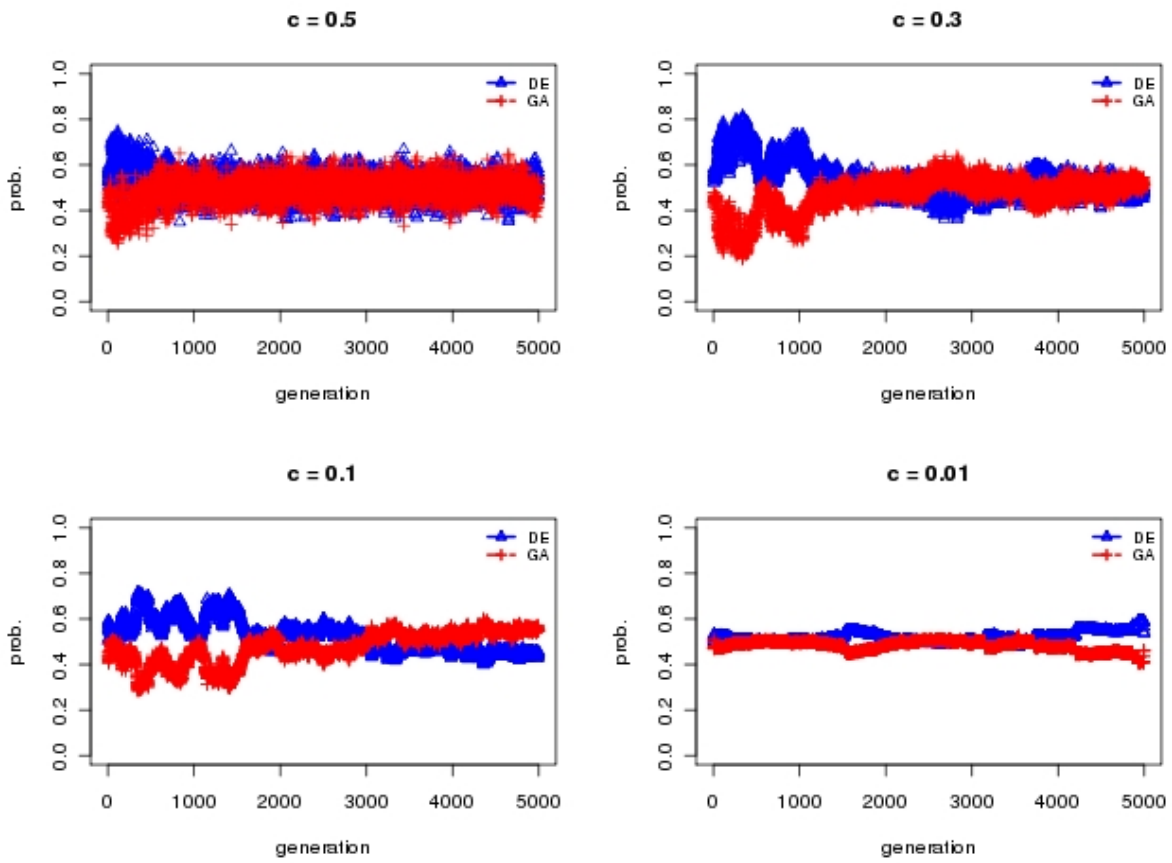

Figure 1: Probabilities of DE and GA during the process for G–01

Function G–01							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	-10.00	-7.05	-3.00	2.22	20	1.00	0.00
GA	-13.39	**-10.53**	-8.02	1.26	20	0.00	1.00
Hybrid (c=0.5)	**-15.00**	-10.25	**-10.00**	1.09	20	0.51	0.49
Hybrid (c=0.4)	-10.00	-9.67	-7.45	0.79	20	0.57	0.43
Hybrid (c=0.3)	-10.00	-9.86	-7.11	0.63	20	0.53	0.47
Hybrid (c=0.2)	-10.00	-9.90	-8.00	0.44	20	0.54	0.46
Hybrid (c=0.1)	-12.07	-10.10	**-10.00**	0.45	20	0.53	0.47
Hybrid (c=0.01)	-13.21	-10.31	**-10.00**	0.93	20	0.52	0.48
Fixed probab.	-12.93	-10.00	-7.00	0.94	20	0.52	0.48
Function G–02							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	-0.164	-0.128	-0.074	0.03	19	1.00	0.00
GA	-0.125	-0.088	-0.067	0.01	20	0.00	1.00
Hybrid (c=0.5)	-0.121	-0.094	-0.066	0.01	20	0.50	0.50
Hybrid (c=0.4)	**-0.165**	-0.146	-0.104	0.01	20	0.80	0.20
Hybrid (c=0.3)	-0.159	-0.147	-0.123	0.01	20	0.84	0.16
Hybrid (c=0.2)	-0.162	**-0.149**	-0.114	0.01	20	0.83	0.17
Hybrid (c=0.1)	-0.157	-0.145	**-0.129**	0.01	20	0.84	0.16
Hybrid (c=0.01)	-0.132	-0.114	-0.089	0.01	20	0.58	0.42
Fixed probab.	-0.128	-0.108	-0.083	0.01	20	0.84	0.16
Function G–03							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	-8.09E-6	-4.04E-7	0	1.76E-6	20	1	0
GA	-9.74E-5	-8.48E-5	-5.15E-5	1.06E-5	20	0	1
Hybrid (c=0.5)	–	–	–	–	–	–	–
Hybrid (c=0.4)	**-1.0E-4**	**-9.33E-5**	-5.04E-5	1.30E-5	20	0.59	0.40
Hybrid (c=0.3)	**-1.0E-4**	-9.15E-5	-8.33E-8	2.32E-5	20	0.59	0.40
Hybrid (c=0.2)	**-1.0E-4**	-8.99E-5	-1.20E-6	2.24E-5	20	0.57	0.42
Hybrid (c=0.1)	**-1.0E-4**	-8.27E-5	-2.14E-7	3.51E-5	20	0.65	0.34
Hybrid (c=0.01)	-1.0E-6	-7.79E-5	-3.21E-9	2.94E-5	20	0.49	0.50
Fixed probab.	-9.92E-5	-9.17E-5	**-7.21E-5**	7.64E-6	20	0.59	0.40

Table 2: Results for G–01, G–02, G–03

Function G–04							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	**-30665.58**	**-30665.58**	**-30665.58**	0.00	20	1.00	0.00
GA	-30367.83	-29575.12	-28973.27	345.56	20	0.00	1.00
Hybrid (c=0.5)	**-30665.58**	-30619.98	-29774.32	194.06	20	0.56	0.44
Hybrid (c=0.4)	**-30665.58**	**-30665.58**	-30665.51	0.01	20	0.61	0.39
Hybrid (c=0.3)	**-30665.58**	**-30665.58**	**-30665.58**	0.00	20	0.61	0.39
Hybrid (c=0.2)	**-30665.58**	-30626.62	-29886.39	169.82	20	0.60	0.40
Hybrid (c=0.1)	**-30665.58**	-30621.31	-29780.14	192.98	20	0.61	0.39
Hybrid (c=0.01)	**-30665.58**	-30522.04	-28870.11	422.72	20	0.61	0.39
Fixed probab.	-30296.32	-29770.84	-28766.01	410.06	20	0.61	0.39
Function G–06							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	**-6961.92**	-6598.19	-4541.61	716.57	16	1.00	0.00
GA	-6960.76	-4193.55	-1493.13	1793.66	20	0.00	1.00
Hybrid (c=0.5)	**-6961.92**	-6455.91	-4491.94	807.33	20	0.60	0.40
Hybrid (c=0.4)	**-6961.92**	-6794.96	-4324.97	586.91	20	0.73	0.27
Hybrid (c=0.2)	**-6961.92**	-6884.69	-5836.10	250.48	20	0.73	0.27
Hybrid (c=0.3)	**-6961.92**	-6668.95	-4274.85	748.64	19	0.79	0.21
Hybrid (c=0.1)	**-6961.92**	**-6957.15**	**-6866.47**	20.80	20	0.75	0.25
Hybrid (c=0.01)	**-6961.92**	-4670.11	-1238.59	2144.49	20	0.54	0.46
Fixed probab.	-6960.97	-5170.21	-1303.37	1671.38	20	0.75	0.25
Function G–07							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	**40.99**	70.71	311.14	80.17	20	1.00	0.00
GA	43.67	124.04	398.91	110.22	20	0.00	1.00
Hybrid (c=0.5)	42.22	69.66	434.37	83.87	20	0.53	0.47
Hybrid (c=0.4)	41.00	71.47	311.14	79.97	20	0.89	0.11
Hybrid (c=0.3)	41.01	72.87	311.14	79.95	20	0.86	0.14
Hybrid (c=0.2)	**40.99**	46.33	59.36	6.55	20	0.88	0.12
Hybrid (c=0.1)	41.00	**43.79**	**49.85**	2.51	20	0.87	0.13
Hybrid (c=0.01)	41.13	62.59	315.55	58.46	20	0.59	0.41
Fixed probab.	42.36	68.93	317.18	57.92	20	0.87	0.13

Table 3: Results for G–04, G–06, G–07

Function G–08							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	**-0.0410**	**-0.0392**	-**0.0231**	0.01	20	1.00	0.00
GA	**-0.0410**	-0.0208	0.0149	0.02	20	0.00	1.00
Hybrid (c=0.5)	**-0.0410**	-0.0274	-0.0001	0.02	20	0.50	0.50
Hybrid (c=0.4)	**-0.0410**	-0.0378	**-0.0231**	0.01	20	0.56	0.44
Hybrid (c=0.3)	**-0.0410**	-0.0385	**-0.0231**	0.01	20	0.57	0.43
Hybrid (c=0.2)	**-0.0410**	-0.0372	**-0.0231**	0.01	20	0.53	0.47
Hybrid (c=0.1)	**-0.0410**	-0.0360	**-0.0231**	0.01	20	0.61	0.39
Hybrid (c=0.01)	**-0.0410**	-0.0288	0.0013	0.02	20	0.49	0.51
Fixed probab.	**-0.0410**	-0.0357	-0.0085	0.01	20	0.57	0.43
Function G–09							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	742.66	910887.85	10000786.58	2874479.12	11	1.00	0.00
GA	708.48	1733.47	8813.44	1698.13	20	0.00	1.00
Hybrid (c=0.5)	719.59	1943.68	10739.05	2360.10	20	0.53	0.47
Hybrid (c=0.4)	686.84	960.85	1946.64	343.22	20	0.90	0.10
Hybrid (c=0.3)	**680.95**	**863.25**	**1482.96**	243.29	20	0.94	0.06
Hybrid (c=0.2)	681.24	1398.70	11105.84	2242.89	20	0.95	0.05
Hybrid (c=0.1)	681.46	986.50	1570.69	288.66	20	0.88	0.12
Hybrid (c=0.01)	725.63	1600.09	8182.44	1607.23	20	0.52	0.48
Fixed probab.	752.66	1402.44	5678.76	1075.04	20	0.94	0.06
Function G–10							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	7048.77	**7303.33**	**8331.48**	374.97	9	1.00	0.00
GA	–	–	–	–	–	–	–
Hybrid (c=0.5)	–	–	–	–	–	–	–
Hybrid (c=0.4)	7048.88	8999.49	15102.05	2645.10	16	0.85	0.15
Hybrid (c=0.3)	**7048.73**	10449.31	21793.63	4379.92	15	0.83	0.17
Hybrid (c=0.2)	7048.77	10414.08	23254.83	5188.52	19	0.85	0.15
Hybrid (c=0.1)	7048.76	12325.22	29941.72	6350.63	17	0.84	0.16
Hybrid (c=0.01)	14857.99	14857.99	14857.99	0.00	1	0.47	0.53
Fixed probab.	–	–	–	–	–	–	–

Table 4: Results for G–08, G–09, G–10

Function G–12							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	-0.986	-0.879	-0.652	0.09	19	1.00	0.00
GA	**-0.999**	-0.995	-0.982	0.00	20	0.00	1.00
Hybrid (c=0.5)	**-0.999**	**-0.998**	**-0.993**	0.00	20	0.47	0.53
Hybrid (c=0.4)	**-0.999**	-0.994	-0.986	0.00	20	0.61	0.39
Hybrid (c=0.3)	**-0.999**	-0.988	-0.950	0.01	20	0.62	0.38
Hybrid (c=0.2)	**-0.999**	-0.992	-0.963	0.01	20	0.58	0.42
Hybrid (c=0.1)	**-0.999**	-0.994	-0.978	0.01	20	0.61	0.39
Hybrid (c=0.01)	**-0.999**	**-0.998**	**-0.993**	0.00	20	0.49	0.51
Fixed probab.	**-0.999**	-0.996	-0.987	0.00	20	0.49	0.51
Function G–18							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	**-0.866**	**-0.633**	-0.499	0.16	20	1.00	0.00
GA	-0.505	-0.419	-0.200	0.08	20	0.00	1.00
Hybrid (c=0.5)	-0.560	-0.485	-0.390	0.03	20	0.50	0.50
Hybrid (c=0.4)	**-0.866**	-0.630	**-0.500**	0.15	20	0.80	0.20
Hybrid (c=0.3)	**-0.866**	-0.630	**-0.500**	0.16	20	0.84	0.16
Hybrid (c=0.2)	**-0.866**	-0.602	**-0.500**	0.13	20	0.79	0.21
Hybrid (c=0.1)	**-0.866**	-0.544	**-0.500**	0.11	20	0.74	0.26
Hybrid (c=0.01)	-0.604	-0.481	-0.087	0.10	20	0.50	0.50
Fixed probab.	-0.649	-0.510	-0.472	0.04	20	0.80	0.20
Function G–19							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	53.85	**3212.25**	**22464.91**	5525.39	20	1.00	0.00
GA	4737.13	18107.21	34655.83	8846.00	20	0.00	1.00
Hybrid (c=0.5)	1398.20	24943.68	43156.52	10595.76	20	0.53	0.47
Hybrid (c=0.4)	57.04	11029.61	31588.97	11313.12	20	0.89	0.11
Hybrid (c=0.3)	86.65	10259.60	34366.06	11410.75	20	0.88	0.12
Hybrid (c=0.2)	75.92	7151.14	33918.37	9724.87	20	0.94	0.06
Hybrid (c=0.1)	**41.23**	11107.58	36609.25	11204.32	20	0.92	0.08
Hybrid (c=0.01)	633.20	17740.93	38723.99	11514.63	20	0.63	0.37
Fixed probab.	2624.95	20572.14	37142.78	10440.43	20	0.94	0.06

Table 5: Results for G–12, G–18, G–19

Function G–21							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	377.67	399.15	420.63	21.48	2	1.00	0.00
GA	–	–	–	–	–	–	–
Hybrid (c=0.5)	–	–	–	–	–	–	–
Hybrid (c=0.4)	245.73	394.41	497.65	79.95	7	0.93	0.07
Hybrid (c=0.3)	292.93	385.97	450.77	47.97	7	0.93	0.07
Hybrid (c=0.2)	276.76	396.22	465.20	62.32	6	0.92	0.08
Hybrid (c=0.1)	**216.54**	406.76	500.69	96.21	7	0.88	0.12
Hybrid (c=0.01)	–	–	–	–	–	–	–
Fixed probab.	382.38	**382.38**	**382.38**	0.00	1	0.93	0.07
Function G–23							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	-390.19	-97.61	0.00	156.84	16	1.00	0.00
GA	–	–	–	–	–	–	–
Hybrid (c=0.5)	-0.00	-0.00	-0.00	0.00	5	0.42	0.58
Hybrid (c=0.4)	**-400.05**	-185.81	**149.95**	205.78	18	0.71	0.29
Hybrid (c=0.3)	-399.97	**-186.29**	**149.95**	184.31	20	0.72	0.28
Hybrid (c=0.2)	-391.00	-167.73	**149.95**	186.53	20	0.71	0.29
Hybrid (c=0.1)	-400.01	-130.60	**149.95**	192.95	20	0.71	0.29
Hybrid (c=0.01)	-192.36	70.20	700.00	233.53	14	0.54	0.46
Fixed probab.	-300.05	56.63	476.35	143.44	19	0.72	0.28
Function G–24							
Algorithm	**Best**	**Avg**	**Worst**	**Std**	**FR**	**pDE**	**pGA**
DE	-3.00	-3.00	**-3.00**	0.00	20	1.00	0.00
GA	-4.87	-3.36	**-3.00**	0.60	20	0.00	1.00
Hybrid (c=0.5)	**-5.51**	-3.69	**-3.00**	0.92	20	0.40	0.60
Hybrid (c=0.4)	**-5.51**	-3.49	**-3.00**	0.94	20	0.39	0.61
Hybrid (c=0.3)	**-5.51**	-3.54	**-3.00**	0.92	20	0.43	0.57
Hybrid (c=0.2)	**-5.51**	-3.53	**-3.00**	0.87	20	0.32	0.68
Hybrid (c=0.1)	**-5.51**	**-3.88**	**-3.00**	1.09	20	0.46	0.54
Hybrid (c=0.01)	**-5.51**	-3.55	**-3.00**	0.90	20	0.50	0.50
Fixed probab.	**-5.51**	-3.74	**-3.00**	0.86	20	0.46	0.54

Table 6: Results for G–21, G–23, G–24

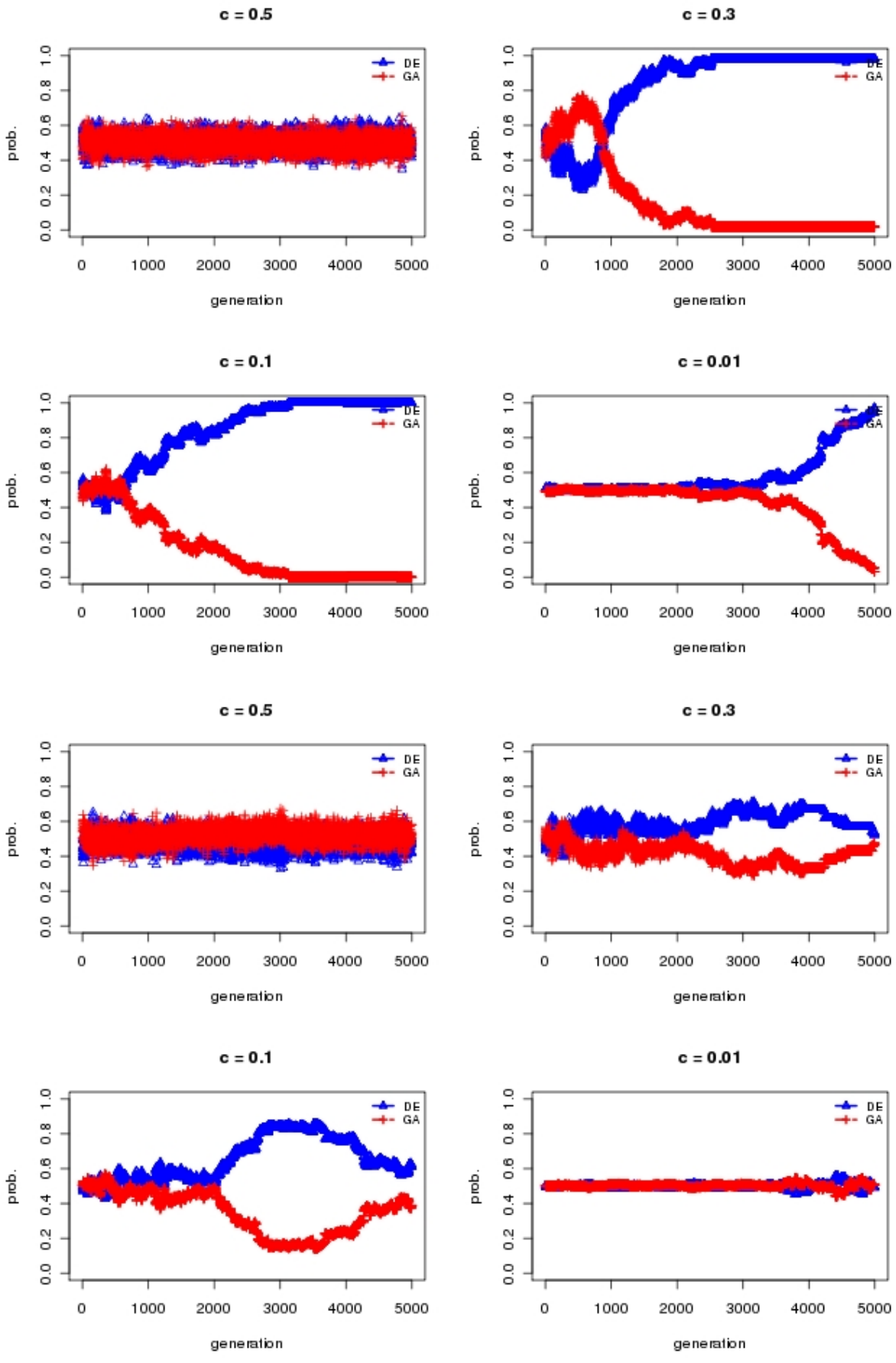

Figure 2: Probabilities of DE and GA during the process for G–02 (upper plots) and G–03 (lower plots).

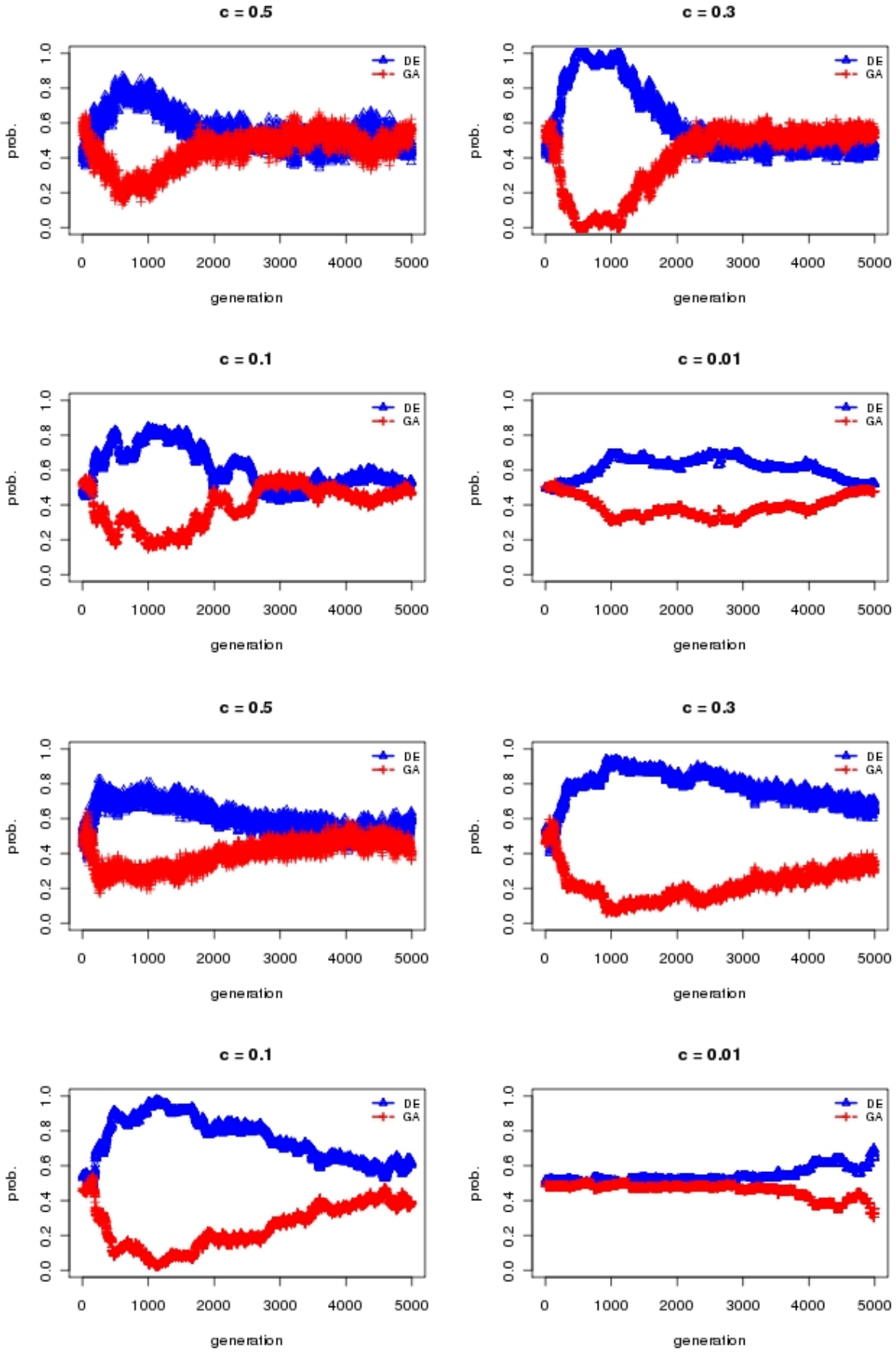

Figure 3: Probabilities of DE and GA during the process for G–04 (upper plots) and G–06 (lower plots).

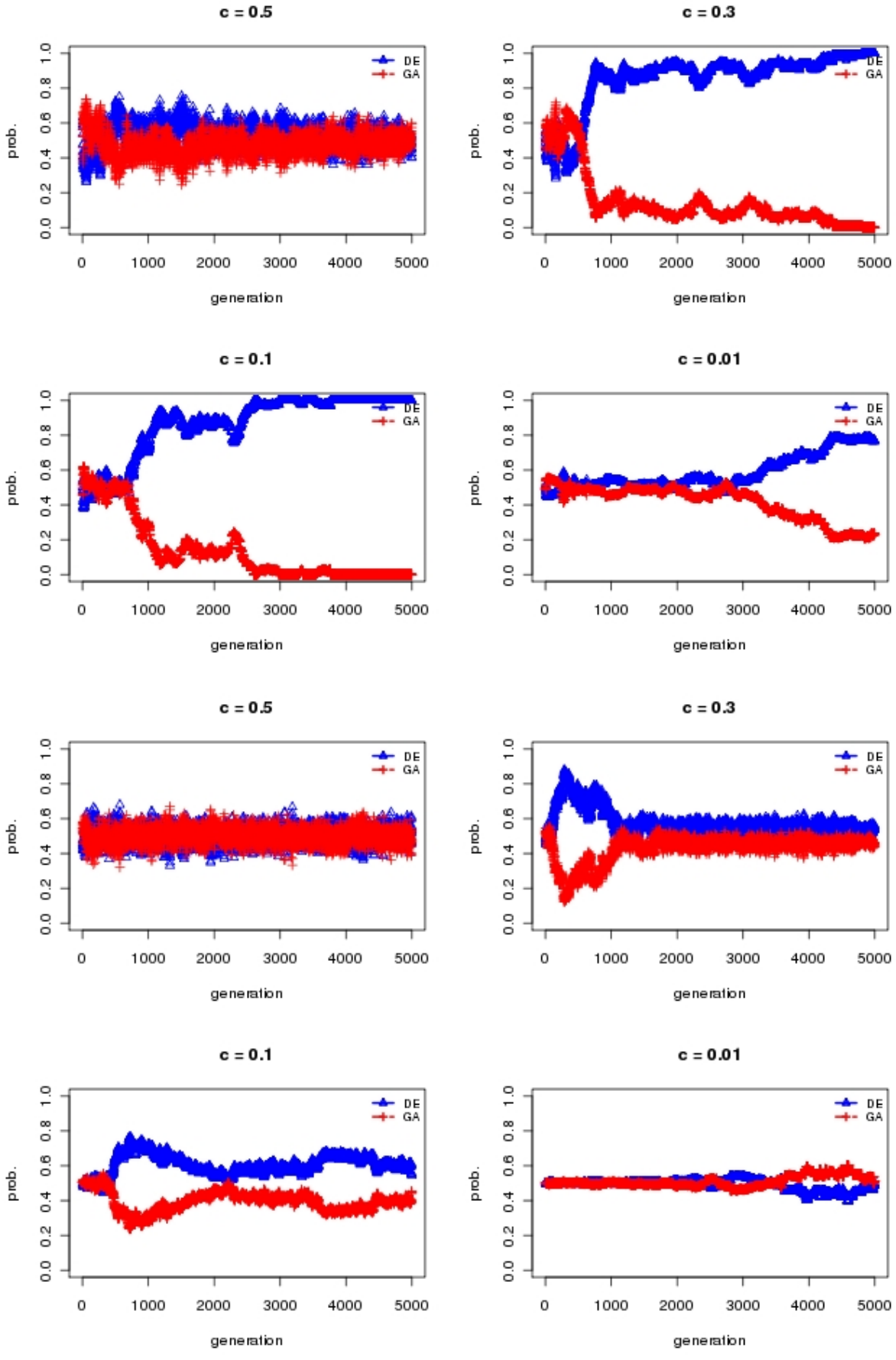

Figure 4: Probabilities of DE and GA during the process for G–07 (upper plots) and G–08 (lower plots).

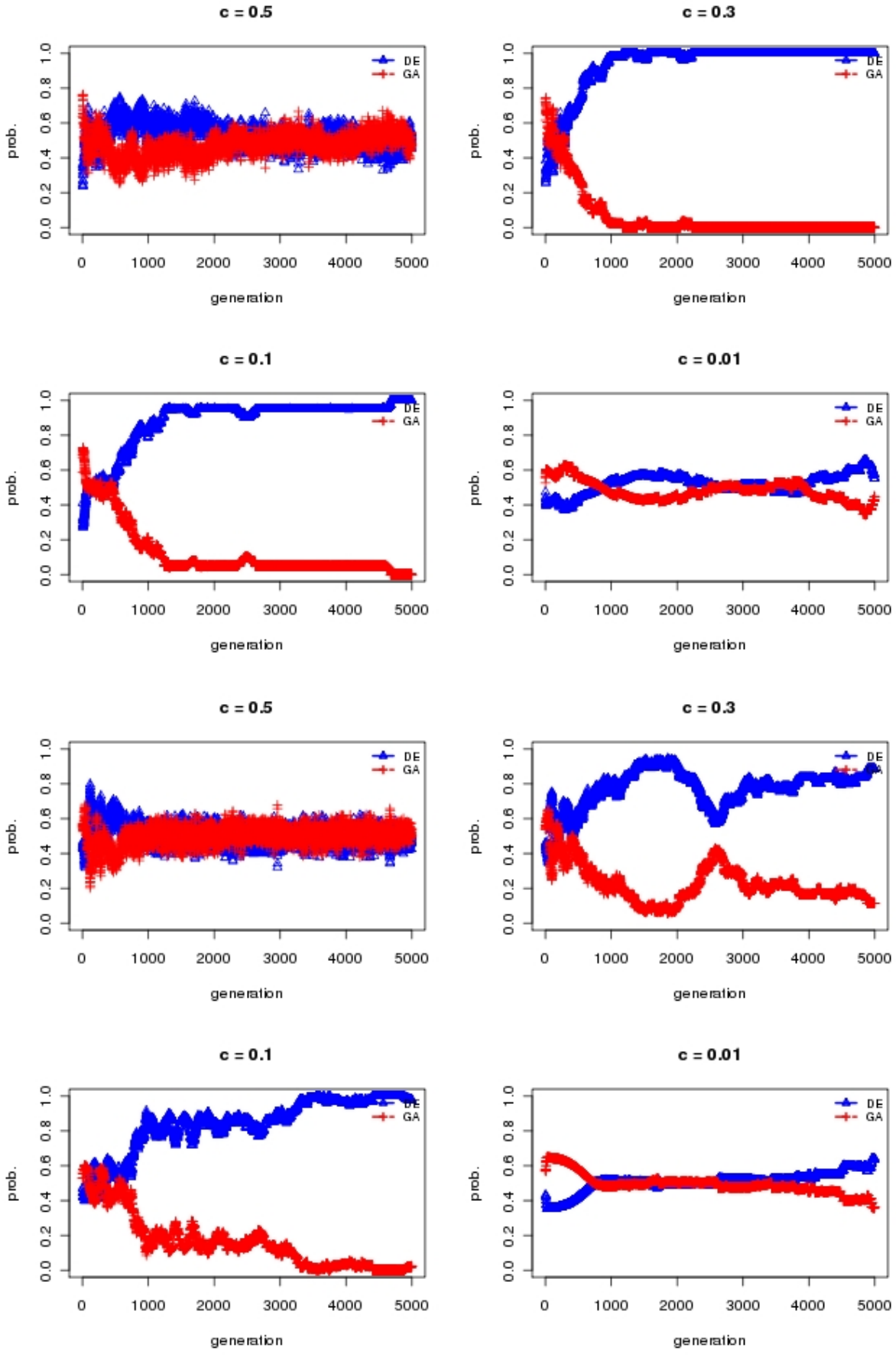

Figure 5: Probabilities of DE and GA during the process for G–09 (upper plots) and G–10 (lower plots).

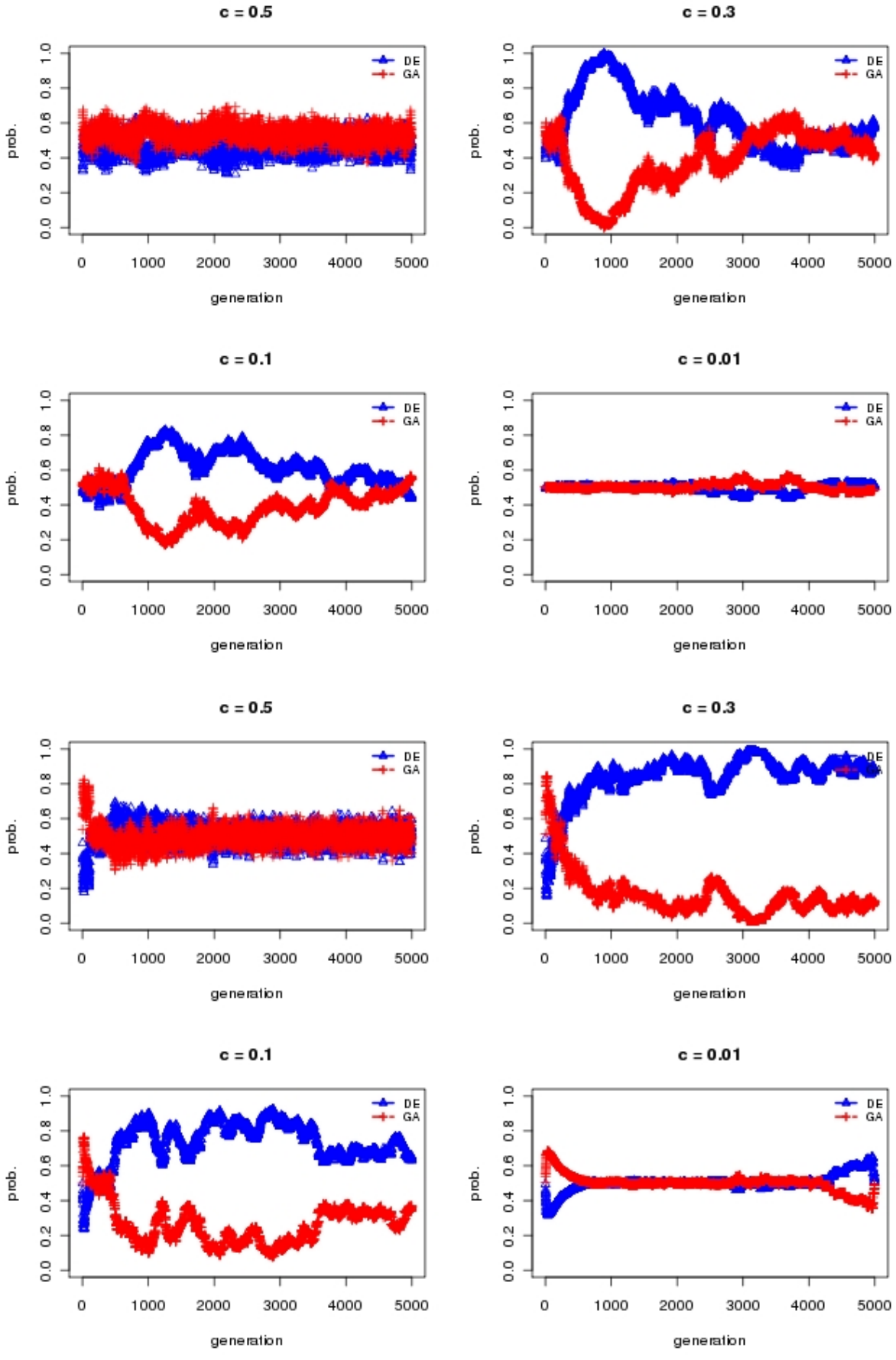

Figure 6: Probabilities of DE and GA during the process for G–12 (upper plots) and G–18 (lower plots).

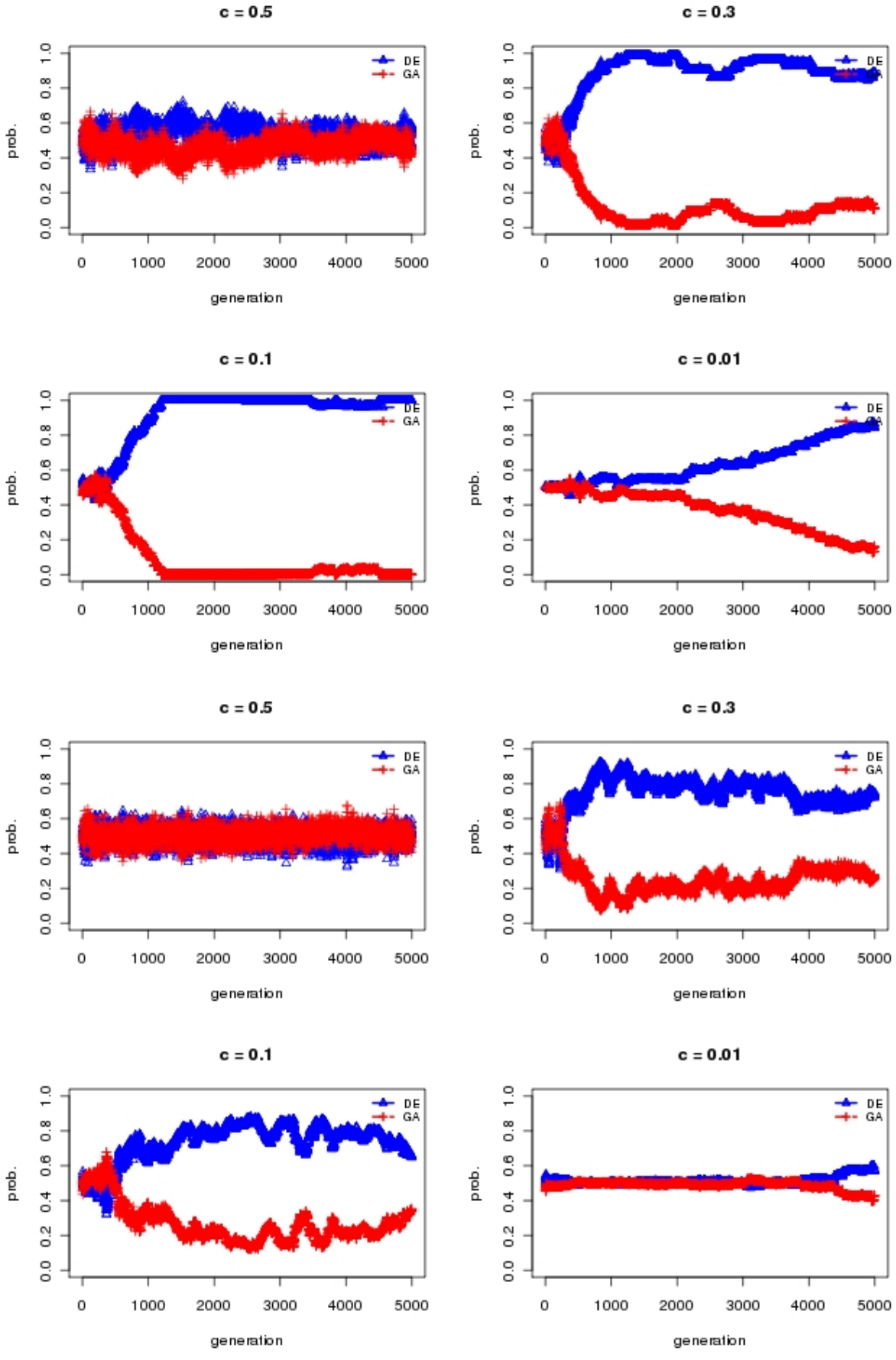

Figure 7: Probabilities of DE and GA during the process for G–19 (upper plots) and G–21 (lower plots).

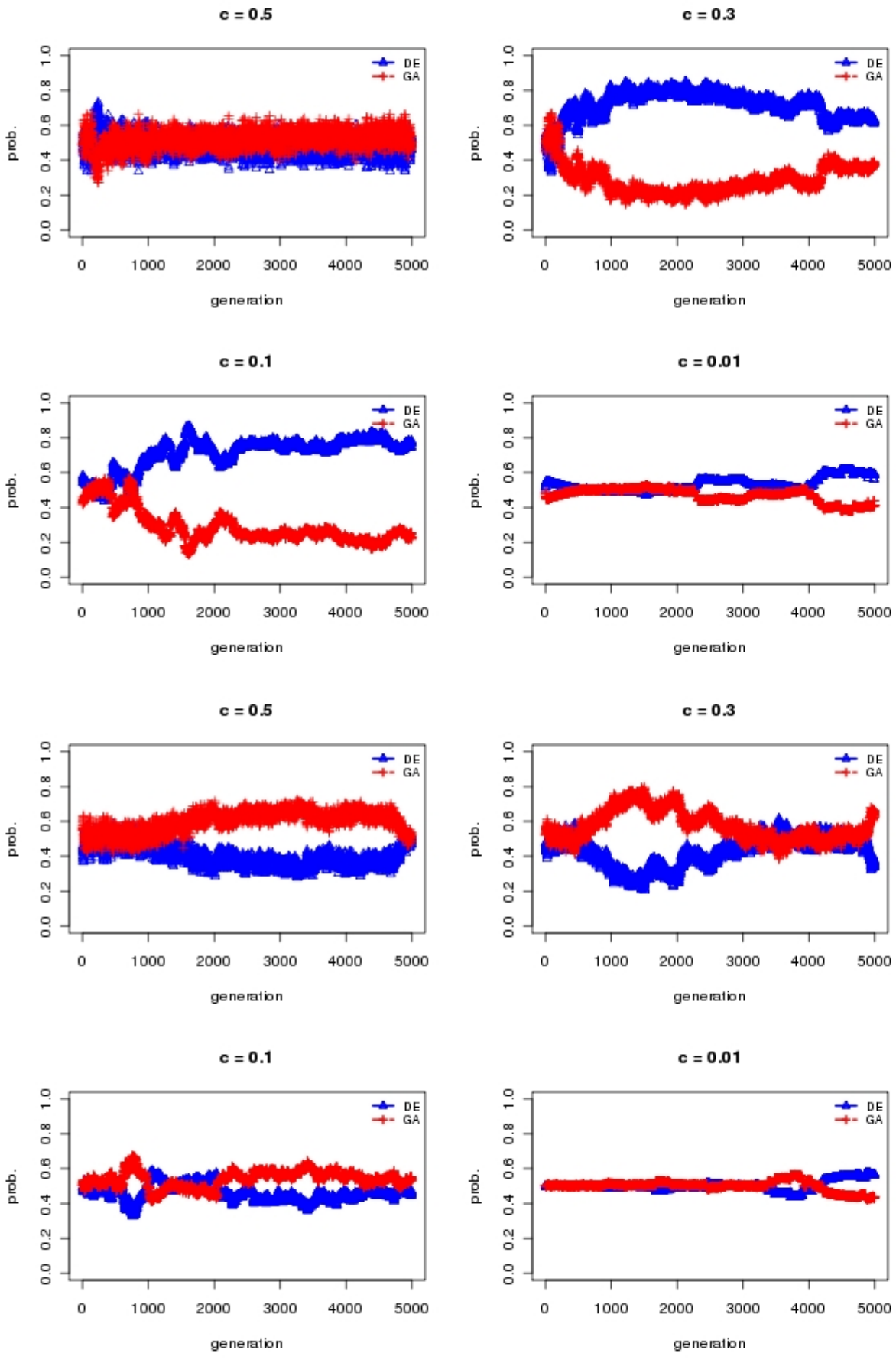

Figure 8: Probabilities of DE and GA during the process for G–23 (upper plots) and G–24 (lower plots).

The most common way of assessing the relative performance of a set V of variants $v_i, i \in \{1, \ldots, n_v\}$ is to define a set F of representative test-functions $f_j, j \in \{1, \ldots, n_f\}$ and then test all variants against all problems measuring the performance $t_{f,v}$ of variant $v \in V$ when applied to function $f \in F$. The performance indicator to be maximized here is the inverse of the minimum objective function value found by variant v in test-function f after 20 runs.

Now a performance ratio can be defined as

$$r_{f,v} = \frac{t_{f,v}}{\min\{t_{f,v} : v \in V\}} \tag{4}$$

It is interesting to be able to assess the performance of the variants in V on a potentially large set of test-functions F in a compact graphical form. This can be attained following Dolan & More (Dolan and Moré, 2002) and defining

$$\rho_v(\tau) = \frac{1}{n_f} |\{f \in F : r_{f,v} \leq \tau\}|$$

where $|.|$ denotes the cardinality of a set. Then $\rho_v(\tau)$ is the probability that the performance ratio $r_{f,v}$ of variant $v \in S$ is within a factor $\tau \geq 1$ of the best possible ratio. If the set F is large and representative of problems yet to be tackled then variants with larger $\rho_s(\tau)$ are to be preferred. The performance profiles thus defined have a number of useful properties (Dolan and Moré, 2002; Barbosa et al., 2010) such as (i) $\rho_v(1)$ is the probability that variant v will provide the best performance in F among all variants in V. If $\rho_{V1}(1) > \rho_{V2}(1)$ then variant $V1$ was the winner in a larger number of problems in F than variant $V2$, and (ii) a measure of the reliability of variant v is its performance ratio in the problem where it performed worst: $R_v = \sup\{\tau : \rho_v(\tau) < 1\}$. As a result, the most reliable variant is the one that minimizes R_v; that is, it presents the best worst performance in the set F:

$$v^* = \arg\min_{v \in V} R_v = \arg\min_{v \in V} \sup\{\tau : \rho_v(\tau) < 1\}$$

It can be seen in figure 9 that (i)the variant (algorithm) Hybrid (0.1) is the most efficient, being the winner in about 80% of the problems, (ii)the variant Hybrid (0.1) is also the most reliable, since it is able to solve all problems within 1.3 times the value obtained by the best solver.

7 CONCLUSIONS

This paper presented a new hybrid and adaptive algorithm to combine Differential Evolution and Genetic Algorithms, two important Evolutionary Algorithms applied in several fields. With this new hybrid algorithm we obtained expressive results when compared with the algorithms separately.

Our proposal reached the best results or the same result that other algorithms in practically all problems with different values for the control parameter c. Moreover, the results show that the best region of c values is [0.1, 0.3].

Another important observation is the behavior of the probabilities during the process with several changes of trajectory. This showed an adjustment of probabilities for the current instant of the search.

The results of this work encourages us to apply the algorithm on other optimization problems of interest.

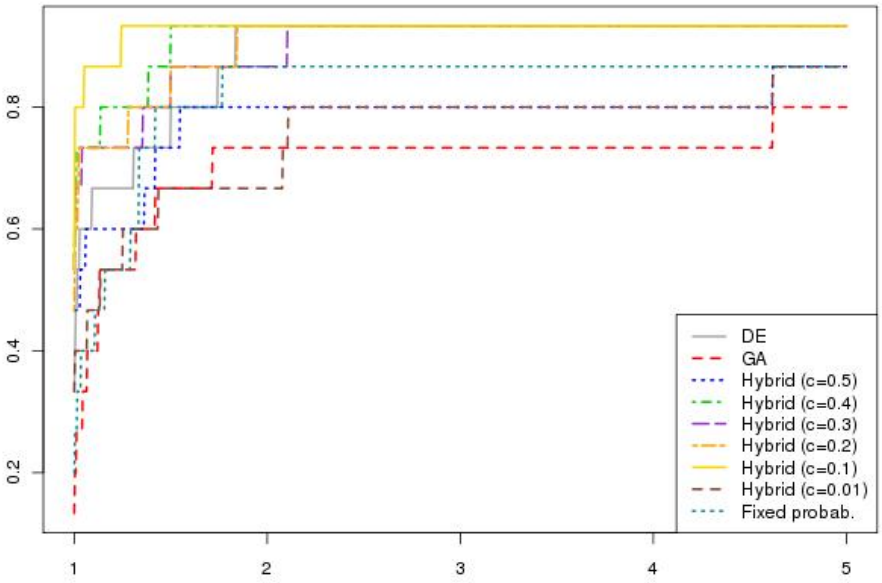

Figure 9: Performance profiles comparing the algorithms.

ACKNOWLEDGMENT

The authors thank CNPq for the support.

REFERENCES

Barbosa H.J.C., Bernardino H.S., and Barreto A.M.S. Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In *IEEE World Congress on Computational Intelligence*. Barcelona, Spain, 2010.

Barbosa H.J.C. and Sá A.M. On adaptive operator probabilities in real coded genetic algorithms. In *SCCC 2000 Workshop on Advances and Trends in Artificial Intelligence for Problem Solving*. Santiago, Chile, 2000.

Deb K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, (186):311–338, 2000.

Deb K. and Agrawal R.B. Simulated binary crossover for continous search space. Technical Report, Indian Institute of Tecnology, 1994.

Dolan E. and Moré J.J. Benchmarcking optimization software with performance profiles. *Math. Programming*, 91:201–213, 2002.

Eshelman L.J. and Schaffer J.D. Real-coded genetic algorithms and interval-schemata. In D.L. Whitley, editor, *Foundation of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann., San Mateo, CA, 1993.

Liang J.J., Runarsson T.P., Mezura-Montes E., Clerc M., Suganthan P.N., Coello C.A.C., and Deb K. Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization. 2006.

Lobo F.G. and Goldberg D.E. A function to test methods applied to global minimization. Technical Report, Illinois Genetic Algorithms Laboratory, 1996.

Mezura-Montes E., Velázquez-Reyes J., and Coelho C.A.C. A comparative study of differential evolution variants for global optimization. *Proceedings of the GECCO'06*, pages 8–12, 2006a.

Mezura-Montes E., Velázquez-Reyes J., and Coelho C.A.C. Modified differential evolution for constrained optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 25–32, 2006b.

Michalewicz Z. *Genetic Algorithms + Data Structures = Evolutionary Programs*. Springer-Velag, New York, 1992.

Price K.V. An introduction to differential evolution. *New Ideas in Optimization*, pages 79–108, 1999.

Qing A. A study on base vector for differential evolution. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008)*, 2008.

Silva E.K., Barbosa H.J.C., and Lemonge A.C.C. An adaptive constraint handling techinique for differential evolution with dynamic use of variants in engineering optimization. *Optimization and Engineering*, pages 1–24, 2010.

Storn R. and Price K.V. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

Whitley D. The genitor algorithm and selective pressure. In *Third International Conference on Genetic Algorithms and their Applications*, pages 116–121. Morgan Kaufmann, San Mateo, CA, 1998.