Is Self-Adaptation of Selection Pressure and Population Size Possible? – a Case Study

A.E. Eiben   M.C. Schut   A.R. de Wilde

Department of Computer Science, Vrije Universiteit Amsterdam
{gusz, schut, ardwilde}@cs.vu.nl

Abstract. In this paper we seek an answer to the following question: Is it possible and rewarding to self-adapt parameters regarding selection and population size in an evolutionary algorithm? The motivation comes from the observation that the majority of the existing EC literature is concerned with (self-)adaptation of variation operators, while there are indications that (self-)adapting selection operators or the population size can be equally or even more rewarding. We approach the question in an empirical manner. We design and execute experiments for comparing the performance increase of a benchmark EA when augmented with self-adaptive control of parameters concerning selection and population size in isolation and in combination. With the necessary caveats regarding the test suite and the particular mechanisms used we observe that self-adapting selection yields the highest benefit (up to 30-40%) in terms of speed.

1 Introduction

Calibrating parameters of evolutionary algorithms (EAs) is a long-standing grand challenge in evolutionary computing (EC). In the early years of the field it was often claimed that Genetic Algorithms (GAs) are not very sensitive to the actual choice of their parameters. Later on this view has changed and the EC community now acknowledges that the right choice of parameter values is essential for good EA performance [8]. This emphasizes the importance of *parameter tuning*, where much experimental work is devoted to finding good values for the parameters *before* the "real" runs and then running the algorithm using these values, which remain fixed during the run. This approach is widely practiced, but it suffers from two very important deficiencies. First, the parameter-performance landscape of any given EA on any given problem instance is highly non-linear with complex interactions among the dimensions (parameters). Therefore, finding high altitude points, i.e., well performing combinations of parameters, is hard. Systematic, exhaustive search is infeasible and there are no proven optimization algorithms for such problems. Second, things are even more complex, because the parameter-performance landscape is not static. It changes over time, since the best value of a parameter depends on the given stage of the search process. In other words, finding (near-)optimal parameter settings is a dynamic optimization problem. This implies that the practice of using constant parameters that do not change during a run is inevitably suboptimal.

Such considerations have directed the attention to mechanisms that would modify the parameter values of an EA on-the-fly. Efforts in this direction are mainly driven by two purposes: the promise of a parameter-free EA and performance improvement. Over the last two decades there have been numerous studies on this subject [8, 10]. The related methods – commonly captured by the umbrella term *parameter control* as opposed to parameter tuning – can further be divided into one of the following three categories. *Deterministic parameter control* takes place when the value of a strategy parameter is altered by some deterministic rule modifying the strategy parameter in a fixed, predetermined (i.e., user-specified) way without using any feedback from the search. Usually, a time-dependent schedule is used. *Adaptive parameter control* works by some form of feedback from the search that serves as input to a heuristic mechanism used to determine the change to the strategy parameter. The important point to note is that the heuristic updating mechanism is externally supplied, rather than being part of the "standard" evolutionary cycle. In the case of *self-adaptive parameter control* the parameters are encoded into the chromosomes and undergo variation with the rest of the chomosome. The better values of these encoded parameters lead to better individuals, which in turn are more likely to survive and produce offspring and hence propagate these better parameter values.

To keep our present investigation feasible we do not want to study on-the-fly adjustment of *all* parameters with *all* parameter control mechanisms. The choice about which combinations to consider is made by the following observations. Of the three options self-adaptation is of particular interest for two reasons. First, it fits the evolutionary framework very smoothly in the sense that the changes to the parameters are evolutionary changes, rather than heuristic ones (deterministic or adaptive control). Second, self-adaptation has a very strong reputation, i.e., overwhelming evidence of being capable of adequate parameter control as shown within Evolution Strategies [3, 17]. This makes us choose for a self-adaptive approach. As for the parameters to be controlled it can be noted that the traditional mainstream of research concentrated on the control of the variation operators, mutation and recombination.

However, there is recent evidence, or at least strong indication, that "tweaking" other EA components can be more rewarding; see for instance [4] showing the relative advantage of controlling the population size, instead of other parameters. This makes us disregard variation parameters and concentrate on parameters for selection and population sizing. The literature on varying population size is rather diverse concerning technical solutions as well as the conclusions regarding how to manage population sizes successfully [2, 9, 12, 14, 6, 13, 16, 18]. The picture regarding the control of selection pressure during a run is more coherent; most researchers agree that the selection pressure should be increasing as the evolutionary process goes on, perhaps a legacy of Boltzmann selection, [1, 20, 7, 15, 11]. In the present investigation we will introduce as little as possible bias towards increasing or decreasing the parameter values.

2 Self-adapting selection pressure and population size

Note that our choice for investigating self-adaptive selection and population sizing implies an interesting challenge. Namely, the parameters regarding selection and population issues (e.g., tournament size or population size) are of global nature. They concern the whole population and cannot be naturally interpreted on local level, i.e., they cannot be defined at the level of individuals like mutation step size in evolution strategies. Self-adaptive population and selection parameters seem to be a contradictory idea. The way we address this challenge is based on making global parameters being derived from local (individual level) parameters via an aggregation mechanism. In this way, the value of a global parameter is determined collectively by the individuals in the population. Technically, our solution is threefold:

1. We assign an extra parameter $p \in [p_{min}, p_{max}]$ to each individual representing the individuals "vote" in the collective decision regarding the given global parameter P.
2. We specify an aggregation mechanism calculating the value of P from the p values in the population.
3. We postulate that the extra parameter p is part of the individual's chromosomes, i.e., an extra gene, and specify mechanisms to mutate these genes.[1]

The aggregation mechanism is rather straightforward. Roughly speaking, the global parameter P will be the sum of the local votes of all individuals p_i calculated as follows:

$$P = \lceil \sum_{i=1}^{N} p_i \rceil \tag{1}$$

where $p_i \in [p_{min}, p_{max}]$, $\lceil \ \rceil$ denotes the ceiling function, and N is the (actual) population size.

Finding an appropriate way to mutate such parameters needs some care. A straightforward option would be the standard self-adaptation mechanism of σ values from Evolution Strategies. However, those σ values are not bounded, while in our case $p \in [p_{min}, p_{max}]$ must hold. We found a solution in the self-adaptive mechanism for mutation rates in GAs as described by Bäck and Schütz [5]. This mechanism is introduced for $p \in (0, 1)$ and it works as follows:

$$p' = \left(1 + \frac{1-p}{p} \cdot e^{-\gamma \cdot N(0,1)}\right)^{-1} \tag{2}$$

where p is the parameter in question and γ is the learning rate which allows for control of the adaptation speed. This mechanism has some desirable properties:

1. Changing $p \in (0, 1)$ yields a $p' \in (0, 1)$.

[1] Note that hereby the parameters in question will be subject to evolution: variation happens through the given mutation mechanisms, while selection is "inherited for free" from the selection upon the hosting individuals.

2. Small changes are more likely than large ones.
3. The expected change of p by repeatedly changing it equals zero (which is desirable, because natural selection should be the only force bringing a direction in the evolution process).
4. Modifying by a factor c occurs with the same probability as a modification by $1/c$.

3 Experimental setup

The test suite[2] for testing GAs is obtained through the Multi-modal Problem Generator of Spears [19]. We generate landscapes of 1, 2, 5, 10, 25, 50, 100, 250, 500 and 1000 binary peaks whose heights are linearly distributed and where the lowest peak is 0.5. The chromosome of each individual consists of 100 binary genes, i.e., $\langle x_1, \ldots, x_{100} \rangle$ and 1 or 2 self-adaptive parameters p (representing the self-adaptation of selection and/or population size).

We use a simple GA, SGA, as benchmark and define 4 self-adaptive GA variants. GASAM is a GA with self-adaptive mutation used as a second benchmark; GASAP and GASAT are the GAs where only one parameter is self-adapted; in GASAPT two parameters are self-adapted simultaneously.

The setup of the SGA is as follows. The model we use is a steady-state GA. Every individual is a 100-bitstring. The recombination operator is uniform crossover; the recombination probability is 0.5. The mutation operator is bitflip; the mutation probability is 0.01. The parent selection is 2-tournament and survival selection is delete-worst-two. The population size is 100. Initialization is random. The termination criterion is $f(x) = 1$ or 10,000 evaluations.

GASAM works on individuals with chromosome $\langle x_1, \ldots, x_{100}, p \rangle$, where p is the self-adaptive mutation parameter. The algorithm works the same as SGA on the bit-part of the chromosome (bitflip), but uses equation 2 for mutation of p.

GASAP works on individuals with chromosome $\langle x_1, \ldots, x_{100}, p_1 \rangle$, where p_1 is the self-adaptive population size parameter. GASAP is different from SGA in that it uses the self-adaptive mechanism from equations 1 and 2 to determine the population size. In this particular case, p is scaled to (0,2) enabling the population to grow as well as shrink. For maintaining enough diversity a lower bound for the population size is enforced.

GASAT works on individuals with chromosome $\langle x_1, \ldots, x_{100}, p_2 \rangle$, where p_2 is the self-adaptive tournament size parameter. GASAT is different from SGA in that it uses the self-adaptive mechanism from equations 1 and 2 to determine the tournament size. For maintaining enough diversity a lower bound for the tournament size is enforced.

GASAPT works on individuals with chromosome $\langle x_1, \ldots, x_{100}, p_1, p_2 \rangle$, where p_1 is the self-adaptive population size parameter and p_2 is the self-adaptive tournament size parameter. GASAPT is a combination of GASAP and GASAT as described above.

[2] The test suite can be obtained from the web-page of the authors.

In all self-adaptive GAs we use $\gamma = 0.22$ according to the recommendation in [5]. The code of the problem instance generator, the particular instances, and all algorithm variants can be obtained from the authors' web-sites.

After 100 runs of each GA, the Success Rate (SR), the Average number of Evaluations to a Solution (AES) and its standard deviation (SDAES), and the Mean Best Fitness (MBF) and its standard deviation (SDMBF) are calculated[3]. The ranking of these measures in forming a judgment about competing EAs is, of course, essential. To this end, it is important that SR and MBF are strongly depend on the maximum number of fitness evaluations M in the termination criterion. In particular, experiments with a lower M typically result in a lower SR and MBF. For AES this link is less strong. It could happen that for a lower M the number of successful runs decreases, but if a run is successful it is of the same length as in the experiments with a higher M. In other words, for a lower M AES could remain the same, while SR and MBF are decreasing. As for us, the speed of an EA that counts most: a faster EA (i.e., an EA with lower AES) is more likely to deliver good performance even for lower values of M.

4 Experimental results and analysis

4.1 Experiments with the self-adaptive scheme

The results of the experiments with the benchmarks GAs and the self-adaptive variants are summarized in Tables 1 and 2 (right hand side, Max=10000 evaluations). Table 1 exhibits the results of the two benchmark EAs, the simple GA (SGA) and the GA with the original self-adaptive mutation GA (GASAM) from [5]. Further to these detailed Tables we offer a graphical representation of the outcomes in Figure 1 (left). Comparing the algorithms it occurs that the differences in terms of SR and MBF are not very big. The curves are crossing and lay rather close to each other. (Note the scale on the y-axis of the MBF graphs in Figure 1.) The AES results are much more discriminating between the GAs. With one exception, the curves are not crossing, implying a consistent ranking based on speed. Also the differences are significant: depending on the problem instance, the best GA outruns the worst one by a factor 1.5 to 3.

Based on the available data we can conclude that the GA with self-adaptive tournament size (GASAT) is the fastest, but the SGA is a close second. Somewhat surprisingly, the GA with self-adaptive mutation rate (GASAM) becoms very slow for the more rugged landscapes and diplays the worst performance in terms of AES. We have performed t-tests with 5% significance level to validate the differences. These tests confirmed that the differences were statistically significant.

[3] For reasons of space, the standard deviation results were omitted. The results are used in the t-tests mentioned later in the paper.

4.2 Additional experiments with an alternative scheme

In addition to the above tests with the self-adaptive GAs we have performed experiments with a modified scheme as well. The reason is that we do have some intuition about the direction of change regarding the parameters. In case of tournament size, if a new individual is better than its parents then it should try to increase selection pressure, assuming that stronger selection will be advantageous for him, giving a reproductive advantage over less fit individuals. In the opposite case, if it is less fit than its parents, then it should try to lower the selection pressure. In case of the population size, this logic might not be applicable, but we do try the idea for both cases. Formally, we keep the aggregation mechanism from equation 1 and use the following rule. If $\langle x, p \rangle$ is an individual, where x is the bitstring and p is the parameter value, to be mutated (either obtained by crossover or just to be reproduced solely by mutation), then first we create x' from x by the regular bitflips, then apply

$$p' = \begin{cases} p + \Delta p & if \quad f(x') \geq f(x) \\ p - \Delta p & otherwise \end{cases} \tag{3}$$

where

$$\Delta p = \left| p - \left(1 + \frac{1-p}{p} e^{-\gamma N(0,1)} \right)^{-1} \right| \tag{4}$$

with $\gamma = 0.22$.

This mechanism differs from "pure" self-adaptation because of the heuristic rule specifying the direction of the change. However, it could be argued that this mechanism is not a clean adaptive scheme (because the initial p values are inherited), nor a clean self-adaptive scheme (because the final p values are influenced by a user defined heuristic), but some hybrid form. In any case, the parameter values represented in a given population undergo regular evolutionary selection because good/bad values survive/disappear depending on the fitness of the hosting individual. For this reason we perceive and name this mechanism *hybrid self-adaptive* (HSA).

We perform extra experiments to see how this alternative usage of the Bäck and Schütz formula effects the performance of the algorithms. Implementing this idea for both the tournament size and the population size yields three new variants GAHSAT, GAHSAP and GAHSAPT with the obvious naming convention. Their results are given in Table 3 (right hand side, Max=10000 evaluations) and Figure 1 (right). They show that the hybrid self-adaptive scheme yields a better control of the tournament size than the pure self-adaptive one, in the sense that the new algorithm GAHSAT outruns GASAT, the winner of the first series of experiments, regarding AES. (Here again we confirmed with a t-test that the differences are significant.) For controlling the population size the effects are exactly the opposite (GASAP beats GAHSAP) and the same holds for the combined algorithm (GASAPT beats GAHSAPT).

At the moment we do not have an explanation for these differences. Nevertheless it is interesting to look at the development of tournament size during a

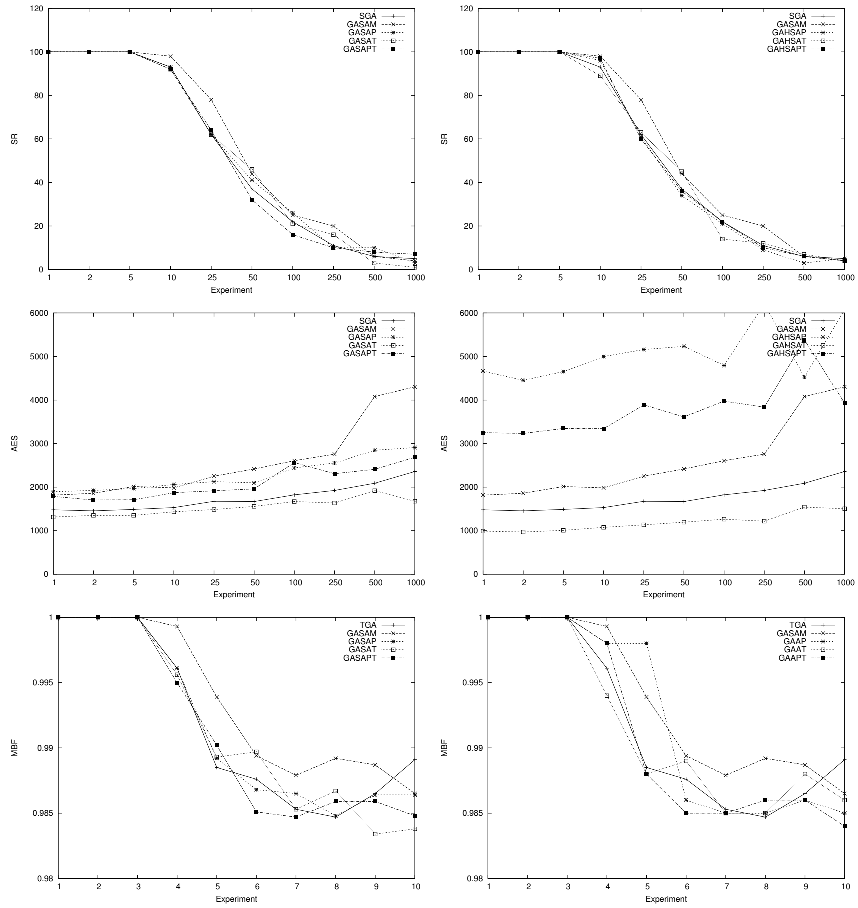

Fig. 1. SR, AES and MBF (from top to bottom) for self-adaptive algorithms (left) and hybrid self-adaptive algorithms (right) with max 10,000 fitness evaluations.

successful run of GASAT or GAHSAT (not shown here because of the lack of space). Such curves show that the selection pressure is growing for quite some time and starts dropping after a while. Intuitively, this is sound, for in the beginning many offspring outperform their parents leading to an increase in their "votes" for the global K. Later on, it becomes increasingly more seldom to produce children better than their parents, which in turn leads to decreasing K. This is also in line with the common view in EC that advocates increasing selection pressure, like in Boltzmann mechanisms. From this perspective we can interpret our results as a confirmation of this common view: the pure self-adaptive variant of the formula from [5] is unbiased w.r.t. the direction of the change and yet it results in increasing increasing tournament size. Last by not least, the overall winner of this contest among 8 GAs is GAHSAT.

The experiments with Max=10000 fitness evaluations give us *one* comparison. However, as our deliberation about the performance measures indicates such outcomes depend on the used maximum. To obtain a second assessment of our GAs we have repeated all experiments with Max=1500. These results are given in the left-hand sides of the tables. They show that the hybrid self-adaptive mechanism is disastrous for GAs where the population size is being modified, while the pure self-adaptive is not. Apparently, the logic that applies to tournament size does not apply to population size. Furthermore, we can see that the margin by which GAHSAT wins from the other becomes bigger than it was for Max=10000. This gives extra support to prefer this algorithm.

5 Conclusions

This investigation provided an answer to our original research question: Is self-adaptation of selection pressure and population size possible and rewarding in an evolutionary algorithm? The answer is double positive. First, we illustrated that it is possible to regulate global parameters (here: tournament size and population size) via aggregating locally specified values. Second, we showed that this can be very rewarding in terms of algorithm performance: the hybrid variant of self-adapting tournament size resulted in a superior GA and the regular self-adaptation variant became second best. Currently we are running additional experiments with constant selection pressure at various levels (tournament size = 2,4,8,16) and with increasing selection pressure (Boltzmann selection) for a broader comparison and more insights in the workings of GA(H)SAT.

Our work also "unearthes" the formula of Bäck and Schütz from [5]. This formula is interesting for its general applicability to mutate bounded parameters and the desirable properties as given in Section 2. The present investigation can also be considered as an assessment of the usefulness of this formula. Without much application specific adjustment we applied it to two parameters and observed that it can greatly improve algorithm performance (e.g., for tournament size). Meanwhile, we also established that it can be harmful (e.g., for population size). Future work is devoted to analyzing why the observed effects occur.

Considering the present investigation from the perspective of the challenge of freeing EAs from (some of) their parameters, our results constitute new evidence that self-adaptation of other than variation operators deserves more attention. We certainly hope that the results and the newly generated questions in this paper will inspire more work in this direction.

References

1. E.H.L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* Wiley, Chichester, UK, 1989.
2. J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS – a genetic algorithm with varying population size. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 73–78. IEEE Press, Piscataway, NJ, 1994.

3. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.
4. T. Bäck, A.E. Eiben, and N.A.L. van der Vaart. An empirical study on GAs "without parameters". In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the 6th Conference on Parallel Problem Solving from Nature*, number 1917 in Lecture Notes in Computer Science, pages 315–324. Springer, Berlin, 2000.
5. Th. Bäck and M. Schütz. Intelligent mutation rate control in canonical genetic algorithms. In Zbigniew W. Ras and Maciej Michalewicz, editors, *Foundations of Intelligent Systems, 9th International Symposium, ISMIS '96, Zakopane, Poland, June 9-13, 1996, Proceedings*, volume 1079 of *Lecture Notes in Computer Science*, pages 158–167. Springer, Berlin, Heidelberg, New York, 1996.
6. J. Costa, R. Tavares, and A. Rosa. An experimental study on dynamic random variation of population size. In *Proc. IEEE Systems, Man and Cybernetics Conf.*, volume 6, pages 607–612, Tokyo, 1999. IEEE Press.
7. Ambedkar Dukkipati, Narsimha Murty Musti, and Shalabh Bhatnagar. Cauchy annealing schedule: An annealing schedule for boltzmann selection scheme in evolutionary algorithms. pages 55–62.
8. A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.
9. A.E. Eiben, E. Marchiori, and V.A. Valko. Evolutionary Algorithms with on-the-fly Population Size Adjustment. In X. Yao et al., editor, *Parallel Problem Solving from Nature, PPSN VIII*, number 3242 in Lecture Notes in Computer Science, pages 41–50. Springer, Berlin, Heidelberg, New York, 2004.
10. A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.
11. Luis Gonzalez and James Cannady. A self-adaptive negative selection approach for anomaly detection. In *2004 Congress on Evolutionary Computation (CEC'2004)*, pages 1561–1568. IEEE Press, Piscataway, NJ, 2004.
12. G.R. Harik and F.G. Lobo. A parameter-less genetic algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 258–265, Orlando, Florida, USA, 1999. Morgan Kaufmann.
13. R. Hinterding, Z. Michalewicz, and T.C. Peachey. Self-adaptive genetic algorithm for numeric functions. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, number 1141 in Lecture Notes in Computer Science, pages 420–429. Springer, Berlin, 1996.
14. F.G. Lobo. *The parameter-less Genetic Algorithm: rational and automated parameter selection for simplified Genetic Algorithm operation*. PhD thesis, Universidade de Lisboa, 2000.
15. Eric Poupaert and Yves Deville. Acceptance driven local search and evolutionary algorithms. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1173–1180. Morgan Kaufmann, 2001.
16. D. Schlierkamp-Voosen and H. Mühlenbein. Adaptation of population sizes by competing subpopulations. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996.
17. H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, 1995.

18. R.E. Smith. Adaptively resizing populations: An algorithm and analysis. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, 1993.
19. W.M. Spears. *Evolutionary Algorithms: the role of mutation and recombination*. Springer, Berlin, Heidelberg, New York, 2000.
20. Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In E.K. Burke X. Yao, J.A. Lozano, J. Smith, J.J.M. Guervos, J.A. Bullinaria, J.E. Rowe, P. Tino, A. Kaban, and H.-P. Schwefel, editors, *PPSN*, number 3242 in Lecture Notes in Computer Science, pages 832–842. Springer, Berlin, Heidelberg, New York, 2004.

	Max=1500 Evaluations						Max=10000 Evaluations					
	SGA			GASAM			SGA			GASAM		
Peaks	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF
1	64	1353	0.9959	21	1405	0.9653	100	1478	1.0	100	1816	1.0
2	59	1361	0.9954	19	1398	0.9712	100	1454	1.0	100	1859	1.0
5	57	1376	0.9953	19	1421	0.9656	100	1488	1.0	100	2014	1.0
10	50	1370	0.9889	9	1375	0.9452	93	1529	0.9961	98	1982	0.9993
25	19	1422	0.9776	1	1166	0.9070	62	1674	0.9885	78	2251	0.9939
50	7	1391	0.9650	0	0	0.8677	37	1668	0.9876	44	2417	0.9894
100	3	1351	0.9590	0	0	0.8449	22	1822	0.9853	25	2606	0.9879
250	0	0	0.9432	0	0	0.7923	11	1923	0.9847	20	2757	0.9892
500	0	0	0.9265	0	0	0.7670	6	2089	0.9865	6	4079	0.9887
1000	0	0	0.9152	0	0	0.7565	5	2358	0.9891	4	4305	0.9865

Table 1. Experimental results of benchmark algorithms in terms of SR, AES and MBF with Max=1500 and Max=10000 fitness evaluations.

	Max=1500 Evaluations									Max=10000 Evaluations								
	GASAP			GASAT			GASAPT			GASAP			GASAT			GASAPT		
Peaks	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF
1	20	1364	0.9806	82	1252	0.9974	41	1322	0.9853	100	1893	1.0	100	1312	1.0	100	1787	1.0
2	27	1428	0.9804	77	1257	0.9974	48	1349	0.9872	100	1925	1.0	100	1350	1.0	100	1703	1.0
5	19	1411	0.9782	73	1244	0.9962	38	1273	0.9842	100	1966	1.0	100	1351	1.0	100	1710	1.0
10	6	1447	0.9672	59	1258	0.9887	31	1300	0.9757	93	2060	0.996	92	1433	0.996	92	1872	0.995
25	6	1416	0.9546	30	1262	0.9817	10	1394	0.9587	62	2125	0.989	62	1485	0.990	64	1915	0.990
50	1	1282	0.9452	13	1279	0.9751	6	1329	0.9519	41	2098	0.987	46	1557	0.990	32	1960	0.985
100	0	0	0.9317	7	1397	0.9687	1	1267	0.9426	26	2341	0.987	21	1669	0.985	16	2563	0.985
250	0	0	0.9279	1	1463	0.9644	0	0	0.9385	10	2554	0.985	16	1635	0.987	10	2307	0.986
500	0	0	0.9131	1	1451	0.9551	0	0	0.9312	10	2846	0.986	3	1918	0.983	8	2410	0.986
1000	0	0	0.9058	0	0	0.9520	0	0	0.9192	3	2908	0.986	1	1675	0.984	7	2685	0.985

Table 2. Experimental results of self-adaptive algorithms in terms of SR, AES and MBF with Max=1500 and Max=10000 fitness evaluations.

	Max=1500 Evaluations									Max=10000 Evaluations								
	GAHSAP			GAHSAT			GAHSAPT			GAHSAP			GAHSAT			GAHSAPT		
Peaks	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF	SR	AES	MBF
1	0	0	0.9273	96	925	0.9996	0	0	0.9508	100	4665	1.0	100	989	1.0	100	3250	1.0
2	0	0	0.9276	97	960	0.9997	0	0	0.9487	100	4453	1.0	100	969	1.0	100	3233	1.0
5	0	0	0.9222	91	982	0.9991	0	0	0.9512	100	4654	1.0	100	1007	1.0	100	3346	1.0
10	0	0	0.9200	82	1026	0.9940	0	0	0.9377	96	4998	0.998	89	1075	0.994	97	3343	0.998
25	0	0	0.9007	55	1065	0.9865	0	0	0.9200	61	5160	0.998	63	1134	0.988	60	3889	0.988
50	0	0	0.8989	37	1146	0.9849	0	0	0.9153	34	5233	0.986	45	1194	0.989	36	3612	0.985
100	0	0	0.8886	26	1152	0.9859	0	0	0.9062	21	4794	0.985	14	1263	0.985	22	3976	0.985
250	0	0	0.8789	11	1161	0.9832	0	0	0.8983	9	6211	0.985	12	1217	0.985	10	3834	0.986
500	0	0	0.8733	3	1437	0.9810	0	0	0.8864	3	4524	0.986	7	1541	0.988	6	5381	0.986
1000	0	0	0.8674	2	1140	0.9785	0	0	0.8768	5	6080	0.985	4	1503	0.986	4	3927	0.984

Table 3. Experimental results of hybrid self-adaptive algorithms in terms of SR, AES and MBF with Max=1500 and Max=10000 fitness evaluations.