A Non-Revisiting Simulated Annealing Algorithm

Shiu Yin Yuen and Chi Kin Chow

Abstract – **In this article, a non-revisiting simulated annealing algorithm (NrSA) is proposed. NrSA is an integration of the non-revisiting scheme and standard simulated annealing (SA). It guarantees that every generated neighbor must not be visited before. This property leads to reduction on the computation cost on evaluating time consuming and expensive objective functions such as surface registration, optimized design and energy management of heating, ventilating and air conditioning systems. Meanwhile, the prevention on function re-evaluation also speeds up the convergence. Furthermore, due to the nature of the non-revisiting scheme, the returned non-revisited solutions from the scheme can be treated as self-adaptive solutions, such that no parametric neighbor picking scheme is involved in NrSA. Thus NrSA can be identified as a parameter-less SA. The simulation results show that NrSA is superior to adaptive SA (ASA) on both uni-modal and multi-modal functions with dimension up to 40. We also illustrate that the overhead and archive size of NrSA are insignificant, so it is practical for real world applications.**

I. Introduction

In simulated annealing (SA) [1], each point **x** in a search space is analogous to a state of some physical system, and the objective function $f(\mathbf{x})$ to be minimized is analogous to the internal energy of the system in that state. The goal of SA is to bring the system, from an arbitrary initial state, to a state with the minimum possible energy.

At each step, the SA heuristic considers some neighbor **x'** of the current solution **x**, and probabilistically decides between moving the system to solution **x'** or staying put in **x**. The probabilities are chosen so that the system ultimately tends to move to solution of lower energy. This probability is governed by the acceptance function $P(.)$:

$$P(\Delta E)=\begin{cases}1 & \text{if } \Delta E<0\\ \exp(-\Delta E/T_a(k_a)) & \text{if } \Delta E\geq 0\end{cases} \quad (1)$$

where $\Delta E = f(\mathbf{x'}) - f(\mathbf{x})$, $T_a(k_a)$ is the acceptance temperature. $P(.)$ describes how the system is able to get out of the local minima. This step is terminated when the solution is good enough for the application, or a given computation budget has been exhausted.

SA simulates the annealing process by a cooling schedule. Initially ka = 0. After every N_g solution generation, $k_a \leftarrow k_a + 1$. One choice for the acceptance temperature is $T_a(k_a) = T_a(k_a) = T_a(0)\lambda^{k_a}$, *where* $T_a(0) = f(\mathbf{x})$ at iteration 0 and λ is a user defined parameter.

Shiu Yin Yuen and Chi Kin Chow are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China (e-mail: {itkelvin, chowchi}@cityu.edu.hk).

The adaptive SA (ASA), an improved version of SA also known as the very fast simulated reannealing [2]-[4], provides significant improvement in convergence speed over standard SA and maintains all the advantages of standard SA. ASA introduced the reannealing concept, in which the acceptance temperature self adjusts according to the fitness history. In addition, it introduced an adaptation to the neighbor picking scheme, to which the distance between the neighbor solution and the current solution is controlled by an adaptive temperature. The details of the reannealing and the neighbor picking are summarized in the following:

Neighbor picking – After every N_g solution generation, k_a and $\{k_i\}$ and are updated as: $k_a \leftarrow k_a + 1$ and $k_i \leftarrow k_i + 1$ for all i.

Suppose $X = \prod[L_i, U_i]$ is a D-dimensional search space (i.e. $i \in [1, D]$) and $\mathbf{x} = [x_1, x_2, \ldots, x_D] \in X$ is the solution of the current iteration, the neighbor of **x**, namely **x'**, is defined as:

$$x_i' = x_i + q_i(U_i - L_i) \text{ and } x_i' \in [L_i, U_i] \text{ for } i \in [1, D] \quad (2)$$

where

$$q_i = \operatorname{sgn}(n_i - 0.5)T_i(k_i)\left(\left(1+\frac{1}{T_i(k_i)}\right)^{|2n_i-1|}-1\right) \quad (3)$$

$$T_i(k_i) = T_i(0)\exp(-ck_i^{\frac{1}{D}}) \quad (4)$$

$$T_a(k_a) = T_a(0)\exp(-ck_a^{\frac{1}{D}}) \quad (5)$$

and n_i is a uniformly distributed random variable in the range [0, 1], c is temperature decay constant and $T_i(0)$ is the initial temperature of x_i at iteration 0.

Reannealing – After every N_a acceptance solutions, $\{T_i\}$ and the acceptance temperature T_a are adjusted according to the local information $f(\mathbf{x})$:

$$T_a(0) \leftarrow T_a(k_a) \quad (6)$$

$$T_a(k_a) \leftarrow f(\mathbf{x}) \quad (7)$$

$$k_a = \left(\frac{1}{c}\ln\left(\frac{T_a(0)}{T_a(k_a)}\right)\right)^D \quad (8)$$

$$T_i(k_i) \leftarrow \frac{\max_{j\in[1,D]} g_j}{g_i} T_i(k_i) \quad (9)$$

$$k_i = \left(\frac{1}{c}\ln\left(\frac{T_i(0)}{T_i(k_i)}\right)\right)^D \quad (10)$$

where $g_i = \left|\frac{f(\mathbf{x}+\mathbf{s_i})-f(\mathbf{x})}{\delta}\right|$; $s_{i,j} = \begin{cases}\delta & j=i\\ 0 & \text{otherwise}\end{cases}$ (11)

978-1-4244-1823-7/08/$25.00

and δ is the step size of $f(.)$.

Though it is claimed in [5] that ASA has good self adaptation ability and its performance is not critically influenced by the specific chosen values of N_g, N_a and c, this claim is only true for the class of Gaussian-Markovian systems and there is no theoretical proof to support that ASA is still robust on other class of systems. In fact, the simulation results in Section IV show that the best fitness found by the ASA is sensitive to the choice of c. One reason is that the parameter c controls the similarity between the current solution and its neighbor (i.e. the search strategy of ASA). To be a truly adaptive SA, the neighbor picking procedure must be *parameter free*. In this article, we propose adopting the non-revisiting scheme suggested by Yuen and Chow [6] to standard SA, which results in a non-revisiting and parameter-less SA – Non-revisiting SA (NrSA). Since all neighbors picked from the search space are guaranteed to be non-revisiting in previous iterations, faster convergence speed is expected. Moreover, NrSA reduces computation cost for a variety of applications [7]-[10]. Furthermore, as the non-revisiting scheme outputs a non-revisiting solution which is closest to the original revisited solution, it acts as a parameter-less neighbor picking scheme that overcomes the selection problem of c in ASA.

This paper is organized as follows: Section II reports a novel SA of which no revisited neighbor solution is generated. Section III presents the simulation setup. Section IV reports the simulations results. A conclusion is drawn in Section V.

II. Non-Revisiting Simulated Annealing

A. Non-Revisiting Scheme

The non-revisiting scheme is proposed by Yuen and Chow in [6]. It is originally applied to genetic algorithm (GA).

Definition 1: Revisits
Suppose Q is a set of evaluated solutions, the solution $\mathbf{x}$ is a revisit if $\mathbf{x} \in Q$. □

The non-revisiting scheme stores all visited solutions $\{\mathbf{s_i}\}$ by a tree-structure archive, namely binary space partitioning (BSP) tree. The search space is partitioned into a set of regions H. The regions are non-overlapping, i.e. $h_i \cap h_j = \varnothing$ for all h_i, $h_j \subset H$ and $i \neq j$, and each $\{h_i\}$ consists of one evaluated solution. In view of the BSP tree, a node represents a region $h \subset H$ and the tree grows along with the iterations. Since the tree construction depends on the sequence of generated solutions, the BSP tree is a random tree and its topology is different from trial to trial.

The scheme is analogous to a black box function (fig. 1). The input $\mathbf{x}$ of that function can be any point in the search space. If $\mathbf{x}$ is a *revisit*, the function outputs a solution $\mathbf{n}$ such that 1) $\mathbf{n} \neq \mathbf{x}$, 2) $\mathbf{x}, \mathbf{n} \in h_x \subset H$ and 3) $\mathbf{n}$ is the nearest neighbor of $\mathbf{x}$. Otherwise, $\mathbf{n}$ is assigned as $\mathbf{x}$.

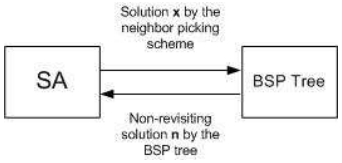

Fig. 1 Communication between SA and the BSP tree.

B. Adaptive neighbor picking of NrSA

Suppose $\mathbf{x}$ is the current solution, the neighbor solution of $\mathbf{x}$; $\mathbf{n} = [n_1, \ldots, n_D]$; is initialized as:

$$n_i = \begin{cases} x_i + \mathrm{sgn}(\eta) \times (U_i \quad L_i) / d & \text{if } i \quad j \\ x_i & \text{otherwise} \end{cases} \tag{12}$$

where $j \in [1, D]$ is a random integer variable, $\eta \in [-1, 1]$ is a uniformly distributed random variable and d is the axis resolution (details can be found in [1]). Afterwards, $\mathbf{n}$ is passed to the non-revisiting scheme. A non-revisiting solution is returned by the scheme if $\mathbf{n}$ is a revisit.

Definition 2: *Nearest neighbor*
In NrSA, the search space S is in the form of a grid; every solution occupies a unique cell in S. Solution $\mathbf{n} \in S$ is a nearest neighbor of solution $\mathbf{x} \in S$ if the cell of $\mathbf{n}$ is adjacent to the cell of $\mathbf{x}$. For a D-dimensional function, every solution, except those at the boundary of the search space, has $2D$ nearest neighbors. □

Suppose $R \subset S$ is a revisit set (revisit zone in S), $B \subset R$ is a subset of R such that every $\mathbf{b} \in \mathbf{B}$ must has at least one nearest neighbor $\mathbf{p}$ which is not a revisit, i.e. $\mathbf{p} \notin R$. The distance between $\mathbf{x}$ and $\mathbf{n}$ must be larger than $|\mathbf{x} - \mathbf{b_0}|$ where

$$\mathbf{b_0} = \arg\min_{\mathbf{b} \in B} |\mathbf{x} - \mathbf{b}| \tag{13}$$

Thus, the search strategy of NrSA (the distance between $\mathbf{n}$ and $\mathbf{x}$) depends on the size of the revisit zone R that $\mathbf{x}$ belongs to. Since the size of R infers the degree of the optimal basin, NrSA adaptively switches between escaping from local optima and fine searching solution. Fig. 2 illustrates an example of the neighbor picking scheme of NrSA. The circles indicate the solution generated from eq. (13) (labeled with $\mathbf{x}$) and the solution returned by the non-revisiting scheme (labeled with $\mathbf{n}$).

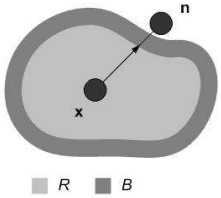

Fig. 2 An example of neighbor picking scheme of NrSA

In the beginning of the search, the number of evaluated solutions (the size of R) is small; **n** is mostly a nearest neighbor of **x**, in which NrSA acts as a random local search. As more solutions are evaluated, the size of R increases; the solution generated from eq. (13) is often a revisit. Thus, the non-revisiting scheme introduces a large jump to **n**, in order to perform a global search.

The neighbor adaptiveness of NrSA on the i^{th} dimension can be represented by the standard deviation of $\{|x_i - n_i|\}$. Fig. 3 shows the adaptiveness of NrSA against iterations on optimizing a 6-dimensional Rastrigin's function (the adaptiveness is indicated as S_N and the details of the Rastrigin's function can be found in Table 3). Fig. 4 shows the fitness of **n** against iterations. Seen from fig. 3, the standard deviations vary along the iterations and they start rising mainly at the 900th, 1600th, 2400th, 2800th and 3400th iteration. Meanwhile, seen from fig. 4, the fitness of **n** increase at those occasions. This observation empirically illustrates that NrSA adaptively pick neighbor solutions according to the fitness of **n**.

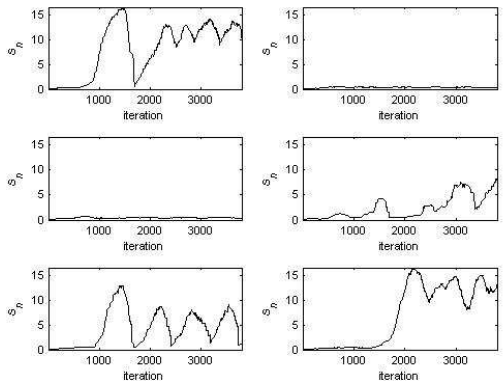

Fig. 3 The actual neighborhood standard deviations of NrSA in a simulation.

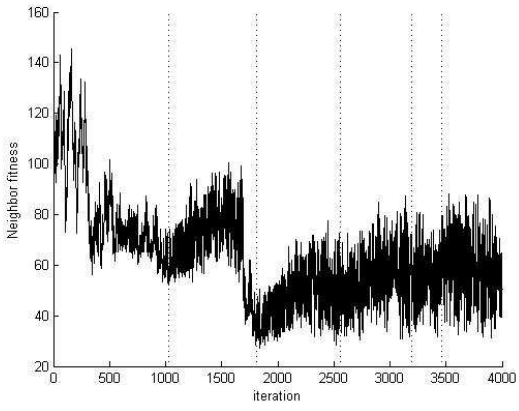

Fig. 4 The fitness of neighbor solutions in a simulation.

C. Mechanism of NrSA

In general, standard SA (SSA) and its variants (i.e. ASA) is possible to be modified as NrSA since the non-revisiting scheme is independent of SA's mechanism; SSA is used in this article. Fig. 5 shows the pseudo code for NrSA. The procedures of NrSA are similar to those of SSA except for the following three extra steps in each iteration:

1. The neighbor solution of **x** is generated from eq. (12).
2. **n** is passed to the BSP tree. A non-revisiting **n** is returned by the tree if the original **n** is a revisit.
3. When **n** is evaluated, it is regarded as a revisit. Thus, the BSP tree should be updated in order to include **n** into the revisit pool.

- Initialize current solution **x**
- Evaluate $\mathbf{x} \rightarrow f(\mathbf{x})$
- Initial the iteration index: $k \leftarrow 0$

while $k < k_{max}$ and $f(\mathbf{x}) > f_{min}$

- Guess **n** by eq. (12)
- Compute **n** in accordance with the non-revisiting scheme.
- Evaluate $\mathbf{n} \rightarrow f(\mathbf{n})$
- Record the sample $[\mathbf{n}, f(\mathbf{n})]$ in the BSP tree
- *if* $P(f(\mathbf{n}) - f(\mathbf{x})) >$ *rand then* $\mathbf{x} \leftarrow \mathbf{n}$
- $k \leftarrow k + 1$

loop

Fig. 5 The pseudo code for Nr SA.

III. Simulation Setup

A. Test function set

In this article, a real valued function set $\mathbf{F} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{14}(\mathbf{x})]$ consisting of 14 functions are employed to illustrate the performance of NrSA. The details (i.e. X – search space, D – function dimension, $\mathbf{x_0}$ – optimal solution, y_0 – optimal fitness) of the functions are listed in Table I. The first six functions are simple unimodal functions whereas the remaining eight functions are multimodal functions designed with a considerable amount of local minima. Meanwhile, the dimensions of the first ten functions are adjustable while the dimensions of the remaining four functions are fixed at two. All functions with the exceptions of f_9, f_{11}, f_{12} and f_{13}, have the global minimum at the origin or very close to the origin. Simulations are carried out to find the global minimum of each function.

B. Setup of test algorithms

To evaluate the impact of the proposed non-revisiting scheme, we compare the optimal fitness found by NrSA with adaptive SA (ASA). For NrSA, the axis resolutions d of the first 10 functions are chosen as 100 whereas the values of d are 2,000 for the remaining four. Moreover, the cooling schedule of NrSA is the same as the standard SA: λ is set to 0.9 in this article. For ASA, the parameters N_a, N_g, c and δ are

chosen as 20, 500, 3 and 0.001 respectively. (this parameters follow those in [11]) $\{T_i(0)\}$ for all i and $T_a(0)$ are initialized as the fitness of the initial **x**.

C. Simulation settings

To provide a fair comparison between the NrSA and the ASA, they are terminated after 40,000 generations in all simulations, i.e. k_{max} = 40,000. All test functions with the exceptions of f_{11}, f_{12}, f_{13} and f_{14}, which is two-dimensional, were tested with dimensions 30 and 40. Since the test algorithms are stochastic, their performance on each test function are concluded by 100 independent runs. All simulations are performed on a PC with 3.2GHz CPU and 1GB memory. The algorithms are implemented in C language.

IV. Simulation Results

Initially, we observed the performance, of NrSA in comparison with ASA in terms of accuracy (quality of the averaged optimal fitness) for 100 trials. To be a practical solution to real world problems, the processing time and the archive size of the BSP tree of NrSA should be within a reasonable range. Thus, the overhead of NrSA related to ASA and the worst case archive size are also observed. Finally, the superiority of the adaptive neighborhood of NrSA over those of ASA is observed for all the functions in dimension of 40. The averaged and the standard deviation (inside brackets) of the optimal fitness for 100 trials are presented in Table II. In Table III, the averaged processing time of NrSA and ASA are presented. Table IV lists the averaged archive sizes for 100 trials. Table V lists the best fitness values found by ASA under different neighborhood standard deviations.

A. Accuracy

Table II lists the average and the standard deviation (inside brackets) of the optimal fitness for 100 trials. The algorithm with bold average fitness value represented that it performs the best among all algorithms. It is clear from the table that NrSA is superior to ASA for the unimodal functions $f_1 - f_6$. It is due to the more robust neighbor picking scheme. NrSA has improved the accuracy significantly when compared with ASA on optimizing multimodal functions: $f_7 - f_{12}$. One reason is that the non-revisiting scheme memorizes all visited local optima, which helps to escape from the local basins. The results of f_{13} and f_{14} obtained from NrSA are competitive with ASA. In general, a consistent performance of NrSA has been observed for all benchmarks considered in this investigation.

B. Processing time

During the search process, an algorithm spends its computation effort on either solution generation or function evaluation. Different algorithms use different strategies to guess a solution, so the corresponding processing time is different. For NrSA, the access to the BSP tree is time consuming and a longer processing time of NrSA compared with other algorithms are expected. As a practical solution to real world applications, the processing time of NrSA should be within an acceptable range. In this section, the computation load of an algorithm, in term of processing time, is studied.

Table III lists the processing time of NrSA and ASA. Though high overhead rates of NrSA are obtained, the maximum overhead amongst all simulation is only 5.0 seconds on using 40,000 fitness evaluations to optimize a 40-dimension function. Furthermore, it is important to point out that, for real world applications such as surface registration [7], optimized design and energy management of heating, ventilating and air conditioning systems [8]-[10], function evaluations are much more time consuming and expensive than solution generation. Thus, the overhead rate of the NrSA is insignificant.

C. Archive size

The variations of the archive size of the BSP tree for 100 trials are observed. The results are listed in Table IV. The archive size is presented as the number of BSP tree nodes. In general, the archive size increases along with the function dimension. One reason is that the increment on function dimension leads to a significant growth on the size of search space, which reduces the chance of tree node pruning. Apart from the increment of function dimension, the nature of an objective function also affects the archive size. The fitness landscape of the Schwefel's problem 2.21 composes of piecewisely flat regions. These regions mislead the solution replacement strategy and NrSA acts more likely a random search. Thus, the chance of tree pruning is reduced. For the Quaric function, the random element introduces many small oscillations (local optimum) into the fitness landscape. In addition, these local optima can hardly be merged to reduce the archive size as they are sparsely distributed on the landscape. The upper bound of the archive size in this test is 40,001. By comparing using a PC that is commonly configured with 1GMB memory, these 40,001 units are just 0.096% of the memory (assuming that each BSP tree node is represented by 24 bytes).

D. Stability of NrSA

In this experiment, we define the stability of an algorithm in terms of quality of the averages and the standard deviation of the optimal fitness under different parameter values (d for the NrSA and c for the ASA). For the first ten test functions, the value of d is varied from 60 to 100. For the remaining four functions, the value of d is varied from 1800 to 2200. The value of c is varied from 1 to 10. The dimensions are fixed at 40 for the first ten test functions. Table IV lists the result for 100 trials.

The results indicate that the effect of d on the optimal fitness is insignificant for all test functions. On the other hand, the choice of c is a critical factor for these functions. In conclusion, it is clear from the results that the averaged

optimal fitness is only slightly dependent on the axis resolution for all the test functions. Therefore, a proper selection of the axis resolution is not a key factor for most problems (at least for the 14 benchmark functions), and the use of the proposed scheme can be identified as a good strategy to overcome the difficulties of selecting a proper mutation parameter.

V. Conclusions

For many real world applications such as surface registration, optimized design and energy management of heating, ventilating and air conditioning systems, the cost of function evaluations, in terms of computation, is very high. To tackle these problems, we propose a novel SA, the non-revisiting SA (NrSA), in which every generated neighbor solution is *novel* – it has not been visited before. The NrSA is an integration of the non-revisiting scheme proposed by Yuen and Chow [6] with a standard SA. Due to the nature of the non-revisiting scheme, the returned non-revisited solution from the scheme can be interpreted as a *parameter free* self-adaptive neighbor picking, and no parametric neighbor picking scheme is required. The NrSA is examined on 14 benchmark unimodal or multimodal functions with dimensions from 30 to 40. The performance of NrSA is illustrated by comparing four quantities: accuracy, processing time, archive size and stability with adaptive SA (ASA). The simulation results show that 1) the best fitness found by NrSA is better than that of ASA; 2) the overhead of NrSA is small; it is practical in real world applications; 3) the archive size, processing time and memory requirement is low even on optimizing a 40-dimensional objective function and 4) a proper selection of the axis resolution is not a key factor for NrSA.

Acknowledgment

The work described in this article was supported by a grant from CityU (7001859).

References

[1] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, Vol 220, no. 4598, pp. 671-680, 1983.

[2] L. Ingber and B. Rosen, "Genetic algorithms and very fast simulated reanneaing: a comparison," *Mathematical and Computer Modelling*, vo1.16, no.11, pp.87-100, 1992.

[3] L. Ingber, "Simulated annealing: practice versus theory," *Mathematical and Computer Modelling*, vol.18, no.11, pp.29-57, 1993.

[4] L. Ingber, "Adaptive simulated annealing (ASA): lessons learned," *J. Control and Cybernetics*, vol. 25, no.1, pp.33-54, 1996.

[5] L. Ingber, "Very fast simulated re-annealing," *Mathl. Comput. Modelling*, vol.12, pp. 967 – 973, 1989.

[6] S. Y. Yuen and C. K. Chow, "A Non-revisiting Genetic Algorithm", in Proc. of *IEEE CEC Conf.*, pp. 4583 – 4590, 2007.

[7] C. K. Chow, H. T. Tsui and T. Lee, "Surface registration using a dynamic genetic algorithm," *Pattern Recognition*, vol. 37, no. 1, pp. 105-117, 2004.

[8] K. F. Fong, V. I. Hanby and T.T. Chow, "HVAC system optimization for energy management by evolutionary programming," *Energy and Buildings*, vol. 38, pp. 220- 231, 2006.

[9] K. F. Fong, T. T. Chow and V.I. Hanby, "Development of optimal design of solar water heating system by using evolutionary algorithm," *Journal of Solar Energy Engineering*, vol. 129, no. 4, pp. 499-501, 2007.

[10] K. F. Fong, "Optimized design and energy management of heating, ventilating and air conditioning systems by evolutionary algorithm," PhD thesis, De Montfort University, UK, 2006.

[11] S. Chen, R. H. Istepanian and B. L. Luk, "Signal processing applications using adaptive simulated annealing," in Proceedings of the 1999 Congress on Evolutionary Computation, vol. 2, pp. 842 – 849, 1999.

TABLE I
THE DETIAILS OF THE FOURTEEN TEST FUNCTIONS

		X	D	$\mathbf{x_0}$	y_0
f_1: Spherical model	$\sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	[30, 40]	[0,…,0]	0
f_2: Schwefel's problem 2.22	$\sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	[30, 40]	[0,…,0]	0
f_3: Schwefel's problem 1.2	$\sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$	[30, 40]	[0,…,0]	0
f_4: Schwefel's problem 2.21	$\max_{i \in [1,D]} \lvert x_i \rvert$	$[-100, 100]^D$	[30, 40]	[0,…,0]	0
f_5: Rosenbrock's function	$\sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i \quad 1)^2 \right]$	$[-29, 31]^D$	[30, 40]	[0,…,0]	0
f_6: Quaric function	$\sum_{i=1}^{D} (ix^4 + random[0,1])$	$[-1.28, 1.25]^D$	[30, 40]	[0,…,0]	0
f_7: Rastrigin's function	$\sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$	$[-5.12, 5.12]^D$	[30, 40]	[0,…,0]	0
f_8: Griewank function	$\frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \frac{x_i}{\sqrt{i}} + 1$	$[-600, 600]^D$	[30, 40]	[0,…,0]	0
f_9: Schwefel's problem 2.26	$-\sum_{i=1}^{D} x_i \sin \sqrt{\lvert x_i \rvert}$	$[-500, 500]^D$	[30, 40]	[420.9687, 420.9687]	$-418.9829D$

f_{10}: Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+e$	[-32, 32][D]	[30, 40]	[0,…,0]	0
f_{11}: Shekel's Foxholes	$\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{i,j})^6}\right]^{-1}$ $\{a_{i,j}\}=\begin{bmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32\end{bmatrix}$	[-98,34]	2	[-32,…,-32]	1
f_{12}: Six-Hump Camel-Back	$4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	[-4.91017, 5.0893] × [-5.7126, 4.2874]	2	[0.08983,-0.7126], [-0.08983,0.7126]	-1.0316285
f_{13}: Branin	$(x_2-\frac{5}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	[-8.142, 6.858] × [-12.275, 2.725]	2	[-3.142,12.275], [3.142,2.275], [9.425,2.425]	0.398
f_{14}: Goldstein-Price	$g(\mathbf{x}) \times h(\mathbf{x})$ *where* $g(\mathbf{x})=1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)$ $h(\mathbf{x})=30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)$	[-2,2] ×[-3,1]	2	[0,-1]	3

TABLE II
THE AVERAGED BEST FITNESS FOUND BY THE NRSA AND THE ASA

function	D	NrSA		ASA		function	D	NrSA		ASA	
f_1	30	**0.000**	(0.000)	111.435	(13.905)	f_7	30	**120.451**	(43.570)	333.066	(42.391)
	40	**0.000**	(0.000)	219.766	(19.723)		40	**219.993**	(43.917)	474.940	(41.731)
f_2	30	**0.000**	(0.000)	3579	(271)	f_8	30	**0.078**	(0.112)	5.010	(0.566)
	40	**0.000**	(0.000)	2.63×10^5	(2.61×10^3)		40	**0.123**	(0.159)	8.864	(0.869)
f_3	30	**1.600**	(2.332)	1282.4	(249.4)	f_9	30	**-10154**	(440.5)	-5703	(637.2)
	40	**2.120**	(2.430)	3654.9	(838.3)		40	**-12382**	(601.4)	-7102	(896.9)
f_4	30	**7.060**	(1.182)	9.021	(1.024)	f_{10}	30	19.220	(2.010)	**10.004**	(6.178)
	40	14.880	(3.374)	**13.353**	(1.740)		40	**19.175**	(0.107)	20.386	(1.175)
f_5	30	**520.4**	(907.2)	2082.5	(656.6)	f_{11}	2	**4.283**	(4.291)	158.022	(229.2)
	40	**437.1**	(772.7)	5656.9	(1891.8)	f_{12}	2	**-0.900**	(0.299)	-0.852	(0.338)
f_6	30	**10.531**	(0.747)	25.733	(4.893)	f_{13}	2	1.289	(3.857)	**0.398**	(0.0)
	40	**15.585**	(1.291)	59.232	(8.552)	f_{14}	2	37.830	(86.677)	**34.861**	(86.601)

TABLE III
THE AVERAGED PROCESSING TIME (IN SEC.) OF THE NRSA AND THE ASA

function	D	NrSA		ASA		function	D	NrSA		ASA	
f_1	30	4.630	(0.688)	0.620	(0.485)	f_7	30	1.490	(0.500)	0.670	(0.470)
	40	7.170	(1.304)	0.820	(0.384)		40	2.050	(0.328)	0.890	(0.313)
f_2	30	3.940	(0.785)	0.410	(0.492)	f_8	30	3.520	(0.793)	0.670	(0.470)
	40	5.880	(1.151)	0.540	(0.498)		40	4.950	(1.169)	0.890	(0.313)
f_3	30	1.880	(0.407)	0.900	(0.300)	f_9	30	1.340	(0.474)	0.480	(0.500)
	40	2.100	(0.300)	1.320	(0.466)		40	1.990	(0.436)	0.630	(0.483)
f_4	30	0.610	(0.488)	0.410	(0.492)	f_{10}	30	1.260	(0.439)	0.680	(0.466)
	40	0.730	(0.444)	0.540	(0.498)		40	1.620	(0.485)	0.890	(0.313)
f_5	30	2.400	(0.529)	1.100	(0.300)	f_{11}	2	1.780	(0.976)	0.410	(0.492)
	40	2.990	(0.360)	1.460	(0.498)	f_{12}	2	3.040	(0.937)	0.080	(0.271)
f_6	30	1.270	(0.444)	0.660	(0.474)	f_{13}	2	3.260	(1.419)	0.070	(0.255)
	40	1.490	(0.500)	0.870	(0.336)	f_{14}	2	3.240	(1.209)	0.070	(0.255)

TABLE VI
THE AVERAGED ARCHIVE SIZE OF THE NRSA

function	D	mean	stdev.	function	D	mean	stdev.	function	D	mean	stdev.
f_1	30	6314	(182)	f_5	30	8433	(1779)	f_9	30	26222	(2627)

	40	9155	(320)		40	12624	(2391)		40	30550	(1793)
f_2	30	7914	(3529)	f_6	30	36864	(2667)	f_{10}	30	35992	(2066)
	40	11965	(6591)		40	39444	(442)		40	37009	(1274)
f_3	30	20617	(2370)	f_7	30	31025	(4665)	f_{11}	2	12481	(8568)
	40	36439	(3054)		40	36617	(1676)	f_{12}	2	7407	(1954)
f_4	30	40001	(0)	f_8	30	11550	(3195)	f_{13}	2	5907	(2134)
	40	40001	(0)		40	16306	(5315)	f_{14}	2	4611	(1358)

TABLE VII
THE STABILIES OF THE NRSA AND THE ASA

function		mean	stdev.	function		mean	stdev.	function		mean	stdev.
f_1	NrSA	**0.00**0	(0.000)	f_6	NrSA	70.11	(10.58)	f_{11}	NrSA	**6.28**	(5.49)
	ASA	598.2	(117.82)		ASA	**19.23**	(2.71)		ASA	226.5	(60.37)
f_2	NrSA	**0.000**	(0.000)	f_7	NrSA	**265.3**	(56.71)	f_{12}	NrSA	**-0.71**	(0.325)
	ASA	6.71×10^5	(3.4×10^3)		ASA	783.1	(137.4)		ASA	-0.4.31	(0.247)
f_3	NrSA	**3.27**	(4.36)	f_8	NrSA	**0.875**	(0.631)	f_{13}	NrSA	1.98	(4.23)
	ASA	58734.2	(1364.3)		ASA	15.64	(5.69)		ASA	**1.387**	(0.617)
f_4	NrSA	**16.84**	(2.11)	f_9	NrSA	**-12087**	(632.4)	f_{14}	NrSA	**40.37**	(89.72)
	ASA	26.87	(11.7)		ASA	-5310	(2891.6)		ASA	45.87	(89.16)
f_5	NrSA	**512.8**	(623.7)	f_{10}	NrSA	**24.37**	(0.257)				
	ASA	6317.6	(998.7)		ASA	48.31	(18.97)				