DIGITAL PATTERN SEARCH AND ITS HYBRIDIZATION WITH GENETIC ALGORITHMS FOR GLOBAL OPTIMIZATION

Nam-Geun Kim, Youngsu Park and Sang Woo Kim
Division of Electrical and Computer Engineering, Pohang University of Science and Technology (POSTETH), Pohang, Korea
kimng@postech.ac.kr, youngsu@postech.ac.kr, swkim@postech.ac.kr

Keywords: Global optimization, Genetic algorithms, Pattern search.

Abstract: In this paper, we present a new evolutionary algorithm called genetic pattern search algorithm (GPSA). The proposed algorithm is closely related to genetic algorithms (GAs) which use binary-coded genes. The main contribution of this paper is to propose a binary-coded pattern called digital pattern which is transformed from the real-coded pattern in general pattern search methods. In addition, we offer a self-adapting genetic algorithm by adopting a digital pattern that modifies the step size and encoding resolution of previous optimization procedures, and chases the optimal pattern's direction. Finally, we compare GPSA with GA in the robustness and performance of optimization. All experiments employ the well-known benchmark functions whose functional values and coordinates of each global minimum have already been reported.

1 INTRODUCTION

Global optimization has attracted much attention recently (Horst and Pardalos, 1995; Pardalos et al., 2000; Pardalos and Romeijn, 2002), because of a wide spectrum of applications in real-world systems. Global optimization refers to finding the extreme value of a given function in a certain feasible region, and such problems are classified in two classes; unconstrained and constrained problems. This paper concerns a class of optimization algorithms that can be applied to bound constrained problems

$$\begin{aligned} \min \quad & f(x) : \mathbf{R}^n \to \mathbf{R}, \\ \text{subject to} \quad & x \in \mathbf{R}^n, \\ & l_i \le x_i \le u_i, \;\; i = 1, \cdots, n, \end{aligned} \tag{1}$$

where l_i, $u_i \in \mathbf{R}$ and $l_i < u_i$.

Although the works that deal with the global optimization are still not enough, they manage to confront the rapid growth of applications. And such work have yielded new practical solvers for global optimization, called meta-heuristics. The structures of meta-heuristics are mainly based on simulating nature and artificial intelligence tools (Osman and Kelly, 1996). Genetic algorithms (GAs) are one of the most efficient meta-heuristics (Goldberg, 1989; Michalewicz, 1996), that have been employed in a large variety of problems. However, most meta-heuristics including GAs suffer from slow convergence that brings about heavy computational costs mainly because they may fail to detect promising search directions, especially in the vicinity of local minima owing to their random constructions.

Combining meta-heuristics with local search methods is a practical solution in overcoming the drawbacks of slow convergence and random constructions of meta-heuristics. In these hybrid methods, local search strategies are included inside meta-heuristics to guide them in the vicinity of local minima, and to overcome their slow convergence especially in the final stage of the search. This paper pursues that approach and proposes a new hybrid algorithm that combines GAs with a new pattern search method. Pattern search methods are a class of direct search methods that require neither explicit nor approximate derivatives. Abstract generalizations of pattern search methods have been provided in (Torczon, 1997; Audet and Dennis, 2003). We will adopt a new idea in pattern search to form a hybrid algorithm. The new pattern search method, called digital pattern search (DPS) method, digitizes the patterns of pattern search methods into binary-codes. Thus, we

can easily combine GAs and pattern search method to construct a global search method called genetic pattern search algorithm (GPSA).

There have been some attempts to utilize the idea of hybridizing local search methods with GA. Simple hybrid methods use the GAs or local search methods to generate the points for new population and then apply other techniques to improve this new population (Günal, 2000; Zentner et al., 2001). Other hybrid methods do some modifications in the GA operations; selection, crossover and mutation using local search methods (Musil et al., 1999; Yang and Douglas, 1998; Yen et al., 1998; Hedar and Fukushima, 2004). However, the method proposed in this paper is different from these hybrid methods in many aspects. One of the main differences lies in the coding representation. We use the DPS methods in which digital patterns are binary-coded genes, and it is capable of using the evolutionary operators in GAs without modifications. Another significant difference is the self-adapting genetic algorithms that modify the step size and chase the approximate optimal direction by using local information from digital patterns. Numerical results from well-known benchmark functions indicate that GPSA exhibits a very promising performance in obtaining the global minima of multimodal functions.

In the remainder of the paper, we briefly review the basics of GAs and pattern search methods in Section 2. Section 3 proposes the DPS methods. The description of the main GPSAs is given in Section 4. In Section 5, we show experimental results. Finally, the conclusion is given in Section 6.

Notation. Let **B**, **R**, **Q** and **Z** denote the sets of binary, real, rational and integer numbers, respectively. All norms will be Euclidean vector norms or the associated operator norm.

2 BACKGROUND

In this section, we will give a brief description of GAs and pattern search methods. Both of them only use the function values rather than derivatives, and they can be used for problems with discrete design parameters. However, they are different in the coding representation. GAs use binary-coded genes, while pattern search methods use real-coded (floating-point) genes. We propose a digital pattern in order to hybridize GAs and pattern search methods.

2.1 Genetic Algorithms

GAs are algorithms that operate on a finite set of points, called a *population*. The population consists of the m bit string $s_{i,m} = [b_{i,m}, \cdots, b_{i,1}]$, where $b \in \mathbf{B}$ and $i \in \{1, \cdots, n\}$, which can be interpreted as the encoding of a vector $x \in \mathbf{R}^n$ for problem (1).

GAs are derived on the principles of natural selection and they incorporate operators for fitness assignment, selection of points for recombination, recombination of points, and mutation of a point.

The pseudo code in Figure 1 describes the steps executed in a general GA.

Randomly generate an initial population $P(0) := \{S_1(0), \cdots, S_\mu(0)\}$ where $S(t) = [s_{n,m}, \cdots, s_{1,m}]$.
Determine the fitness of each individual.
Repeat $t = 1, 2, \cdots$
  Perform recombine with probability p_r.
  Perform mutation with probability p_m.
  Determine the fitness of each individual.
  Perform replacement with an elitist replacement policy.
Until some stopping criterion is satisfied.

Figure 1: Pseudo code of general GA.

GAs start by generating an initial population $P(0)$ of μ randomly generated points $S(0)$. Then, the fitness values are evaluated for each point in $P(0)$. The fitness of a point indicates the worth of the point in relation to all other points in the population. The selected points are recombined to a new pair of points. In recombination, the crossover position is randomly selected with a probability of $p_r \in [0,1]$ and the bits after this position are exchanged between the two points. Each recombined point is mutated by a mutation, which changes the value of some bits of the binary strings with a probability of $p_m \in [0,1]$. Afterwards, replacement selects the μ_e fittest points ($0 < \mu_e < \mu$) of the generation as the elite set. These points will be put in the next generation.

2.2 Pattern Search Methods

According to (Audet and Dennis, 2003), pattern search methods have common things after a finite number of iteration. They search for a cost function value lower than that of the current iterate x_k on the trial points in the *poll set*

$$L_k = \{x_k + \Delta_k p_k,\ p_k \in P_k\}, \tag{2}$$

where $\Delta_k > 0$ is a step size, and a *pattern* p_k is the columns of the *pattern matrix* P_k defined in (Torczon, 1997). The pattern matrix is decomposed into a basis matrix $B \in \mathbf{R}^{n\times n}$ and a generating matrix $C_k \in \mathbf{Z}^{n\times p}$, $p > 2n$. Restrictions on C_k guarantee that the columns of BC_k span $\mathbf{R}^n$. Conceptually, the generating matrix

defines the search directions, while the basis matrix rotates and scales the search directions to determine the coordinate system used during the search.

In addition, each PS method has a rule called *search step* (Audet and Dennis, 2003) that selects a finite number of points on a *mesh* defined by

$$M_k = \{x_k + \Delta_k P_k z,\ z \in \mathbf{Z}^p\}. \quad (3)$$

At iteration k, the mesh is centered around the current iterate x_k, and its fineness is parameterized through the step size Δ_k. The search step strategy that gives the set of points is usually provided by the user; it must be finite and the set can be empty.

The pseudo code in Figure 2 describes the main elements of a pattern search method. It is based on the method presented in (Audet and Dennis, 2003).

Let the initial solution $x_0 \in \mathbf{R}^n$ and step length Δ_0 be given.
Repeat $k = 1, 2, \cdots$
  Perform the **Search Step**:
    Evaluate f on a finite subset of trial points on the mesh M_k defined by (3).
  Perform the **Poll Step**:
    Evaluate f on the poll set defined by (2).
  Update the pattern matrix and Δ_k.
Until some stopping criterion is satisfied.

Figure 2: Pseudo code of pattern search method.

The scenario of a pattern search method starts with fitting the initial solution, and then two search stages are invoked. The first staged is a *search step* in which any search procedure can be defined by the user to generate trial points from M_k. The main role of the *search step* is to achieve faster convergence of pattern search method. The other stage called *poll step* is performed as a systematic search in order to exploit a region around the current solution. If the *search step* and *poll step* fail to produce a trial step that gives a simple decrease, then the step size is reduced to refine the mesh. Otherwise, the step size is increased or preserved. The pattern search method may be terminated when the step size becomes small enough.

3 DIGITAL PATTERN SEARCH METHOD

In this section, we propose the digital pattern search (DPS) method before introducing genetic pattern search algorithm (GPSA). This section formulates the abstraction of DPS methods. The definitions and the algorithm for generalized DPS methods follow descriptions of pattern search methods provided in (Torczon, 1997) and (Audet and Dennis, 2003).

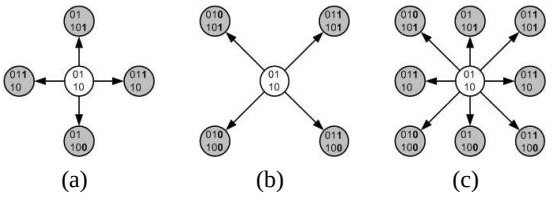

Figure 3: Trial points generated by some kinds of digital patterns. Ash-colored bulbs are trial points. (a), (b), and (c) correspond to digital patterns of compass search, evolutionary operation using factorial designs and coordinate search, respectively.

3.1 Digital Pattern

Adding 0 or 1 to an existing binary string as a least significant bit (LSB) can be interpreted as generating trial points of pattern search methods. The decoded real number of a new binary string that is appended LSB "0" is decreased and that is appended LSB "1" is increased than that of the original binary string. This property is adopted as pattern to generate trial points which are solution candidates. Figure 3 depicts the 2 bit string's trial points that are presented by binary strings or binary matrices in which each row represents an encoded string for the corresponding parameter.

Pattern search methods can be divided by pattern matrix P_k into a compass search, evolutionary operation, coordinate search, and so on (Torczon, 1997). Digital pattern can describe some kinds of patterns according to whether or not an LSB is attached to each dimension of bit strings. Figure 3 depicts some kinds of digital patterns that mimic pattern of compass search, pattern of evolutionary operation using factorial designs, and pattern of coordinate search, respectively.

The DPS method requires a mechanism for decoding from each m-bit string $s_{i,m} = [b_{i,m}, \cdots, b_{i,1}]$ to the corresponding object variable $x_{i,m}$. According to the standard binary decoding function $f_d : \{0,1\}^m \rightarrow [u_i, v_i]$, where (Michalewicz, 1996), the real value is

$$x_{i,m} = f_d(s_{i,m}) = u_i + \frac{v_i - u_i}{2^m - 1} \sum_{j=0}^{m-1} b_{i,(j+1)} 2^j. \quad (4)$$

If 0 is added to the LSB of $s_{i,m}$, let the new string be termed a 0-*bit child string*, $s^0_{i,m+1}$. On the other hand, if 1 is added, it is termed a 1-*bit child string*,

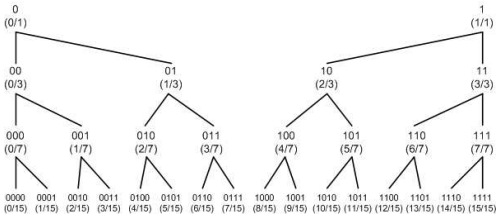

Figure 4: Biased binary tree structure together with corresponding normalized real values in parenthesis.

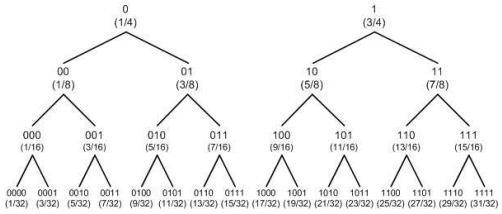

Figure 5: Unbiased binary tree structure together with corresponding normalized real values in parenthesis.

$s^1_{i,m+1}$. And the real values of the child strings are

$$x^0_{i,m+1} = \frac{u_i}{2^{m+1}-1} + \frac{2^{m+1}-2}{2^{m+1}-1}\, x_{i,m},$$
$$x^1_{i,m+1} = \frac{v_i}{2^{m+1}-1} + \frac{2^{m+1}-2}{2^{m+1}-1}\, x_{i,m}.$$

The Difference between a parent string and each child string implies the step size Δ_k in pattern search methods. For a given parent string $x_{i,m}$, the comparison between the distances of $|x^0_{i,m+1} - x_{i,m}|$ and $|x^1_{i,m+1} - x_{i,m}|$ is given by

$$| x^1_{i,m+1} - x_{i,m} | - | x^0_{i,m+1} - x_{i,m} | = \frac{(v_i - u_i) - 2x_{i,m}}{2^{m+1}-1}. \quad (5)$$

From (5), the differences between two step sizes vary according to the position of the parent string in the finite intervals $[u_i, v_i]$.

In Figure 4, the property in (5) is clearly visualized in the form of binary trees whose nodes are represented by binary strings and their corresponding real numbers. Owing to the specific property of digital pattern that increases the bit length of binary strings, the standard binary decoding function (4) has a biased search tendency to incline its steps toward the middle point of the finite intervals $[u_i, v_i]$ depending on the location of a parent string. Therefore, we need a more suitable binary decoding function for digital pattern and the *unbiased binary decoding function* is designed as:

$$x_{i,m} = u_i + \frac{v_i - u_i}{2^{m+1}} \left(\sum_{j=0}^{m-1} b_j 2^{j+1} + 1 \right),$$

and the real values of the child strings are given as:

$$x^0_{i,m+1} = x_{i,m} - \frac{v_i - u_i}{2^{m+2}}, \;\; x^1_{i,m+1} = x_{i,m} + \frac{v_i - u_i}{2^{m+2}}. \quad (6)$$

Both step sizes are the same, $(v_i - u_i)/\, 2^{m+2}$. Figure 5 shows that the unbiased binary decoding function guarantees the symmetric search property.

To define digital pattern, we treat bit strings with the decoded real value defined in the iterative form (6). According to (6), the real values of trial points are analogous to the description of generalized pattern search methods in (Torczon, 1997). A basis matrix can be defined a nonsingular matrix $B \in \mathbf{R}^{n \times n}$

$$B = \mathrm{diag}(l_1, \ldots, l_n) = \mathrm{diag}(u_1 - v_1, \ldots, u_n - v_n),$$

where $\mathrm{diag}(\cdot)$ is a diagonal matrix. B represents the intervals of each dimension of x. A generating matrix $C \in \mathbf{Z}^{n \times p}$, where $p > 2n$, contains in its columns combinations of $\{1, 0, -1\}$, except for the column of zeros. For example, when digital pattern for coordinate search is executed for $n = 2$, we have a generating matrix such as

$$C = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 \\ 0 & 1 & 0 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}.$$

It can be seen in Figure 3 (c).

Digital pattern p is then defined by the columns of the digital pattern matrix $P = BC$. Because both B and C have the rank n, the columns of P span $\mathbf{R}^n$. The step size Δ_m is defined as $\Delta_m = \frac{1}{2^m}$ under the given bit string length m. Thus the poll set composed of points neighboring the current x_m in the directions of the columns of C is expressed as

$$L_m = \{ x_m + \frac{1}{4}\Delta_m p, \; p = Bc \text{ and } c \in C \}.$$

Among them, the best one is chosen by evaluation as an optimal solution of L_m, $x^*_{m+1} = x_m + \frac{1}{4}\Delta_m B c^*_{m+1}$, where c^*_{m+1} is the column of C as a *direction vector* pointing the search direction toward the optimal solution.

3.2 Digital Step

Using the standard binary representation, the DPS method can get caught on a "Hamming cliff", being confined to the barrier of binary branches. For example, if the DPS method is started at the highest node of "0" in the tree, it can never escape from the left half plane of parameters. *Digital step* in the DPS method is employed to avoid such problems. If a binary string undergoes *increment addition* (INC)

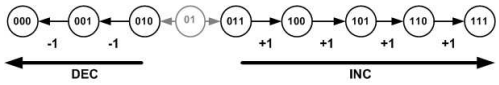

Figure 6: Process of *digital step.* in one dimensional diagram.

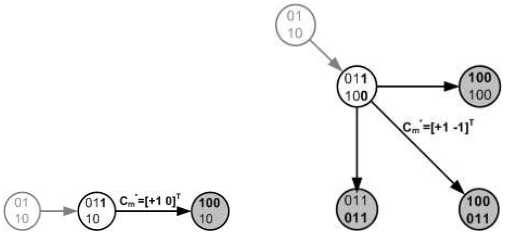

Figure 7: Process of *digital step.* in two dimensional diagram. (a) and (b) correspond to the results depended on c_m^*, $[+1\ 0]^T$ and $[+1\ -1]^T$, respectively.

or *decrement subtraction* (DEC), the real number of each processed string increases or decreases. Through the simple operations of INC and DEC for binary strings, *digital step* can readily remove the barrier between any binary trees and broadly search the effective area that has a high possibility of finding the optimal solution. Figure 6 illustrates the process of *digital step* (if direction vector is 1, then INC is executed, otherwise DEC is achieved).

In *digital step*, the objective function f is evaluated at a finite number of points on a mesh to try to find a point that yields a lower objective function value than the current point. The basic component in the definition of *digital step* is the mesh. The mesh is a discrete subset of $\mathbf{R}^n$ whose fineness is parameterized by the step size Δ_m as follows:

$$M_m = \{x_m^* + \Delta_m \mathrm{diag}(z) B c_m^* : z \in \mathbf{Z}^n\},$$

where $\mathrm{diag}(\cdot)$ is a diagonal matrix, z is the vector of nonnegative integers and c_m^* is a direction vector chosen by evaluation of poll set. This way of describing the mesh is different from the form in (Audet and Dennis, 2003). This specific sub-technique of the DPS method attempts to accelerate the progress of the algorithm by exploiting the information gained from the search.

The trial points generated by *digital step* are decided by a sort of digital patterns. Figure 7 shows the examples of *digital step* in a two dimensional search space. INC or DEC is selected by c_m^* from the previous digital pattern. Then *digital step* continues to search in the trial points which are generated by performing INC or DEC at each dimension. Generation of the trial points by the digital pattern is executed one time, while generation of the trial points by *digital step* continues until a local minimum is attained on the same bit mesh.

3.3 Digital Pattern Search Method

The pseudo code in Figure 8 describes the proposed digital pattern search method.

Set an initial row length α and a final row length β.
Randomly generate an initial binary matrix $\Gamma_{n\times\alpha}$.
for each low length $m = \alpha : \beta$ **do**
1. Perform the **Digital Pattern**.
2. Evaluate trial points and determine a direction vector.
3. **while** (a better solution is attained) **do**
   (a) Perform the **Digital Step**.
   (b) Evaluate trial points.
   end while

end for

Figure 8: Pseudo code of digital pattern search method.

The basic structure of the DPS method consists of two asynchronous loops. The outer loop (steps 1-3) selects the best trial point generated by digital pattern and hands over a direction vector to *digital step*. The inner loop (step 3) conducts finite searches in the guided direction vector until the consecutive *digital step* fails to make progress. When this occurs a local minimum of a session is found; the inner loop then terminates and the outer loop starts the next session. At steps 1 and 3(a), digital pattern and *digital step* generate $n \times m$ binary matrices as trial points, i.e., one of the matrices $\Gamma_{n\times m}$ implies that each row n represents an encoded string for the corresponding parameter, and that the row length m is exponentially proportional to the resolution of parameters. To evaluate trial points in steps 2 and 3(b), each row of Γ needs to be converted to the real number using a binary decoding function. After evaluation in step 2, the least significant column of the best trial point is appointed the direction vector which is handed over to *digital step* for further exploration.

4 GENETIC PATTERN SEARCH ALGORITHM

GPSA uses the main operations of GA; recombination, mutation, and replacement, on a population to encourage the exploration process. Moreover, the GPSA tries to improve the new children by applying

DPS method. Figure 9 shows the pseudo code describing GPSA.

```
Set an initial row length α and a final row length β.
Set an GA's generation number N.
Randomly generate an initial population P_α(0) :=
{Γ^1_{n×α}(0), ··· , Γ^μ_{n×α}(0)}
Determine the fitness of each individual.
for each low length m = α : β do
  for GA's generation k = 0 : N do
    Perform recombine with probability p_r.
    Perform mutation with probability p_m.
    Compute the fitness of each individual.
    Perform replacement with an elitist replacement policy.
  end for
  Perform the Digital Pattern.
  Compute the fitness of trial points of each individual and determine a direction vector.
  while a better solution is attained do
    Perform the Digital Step.
    Compute the fitness of trial points of each individual.
  end while
  Perform replacement with an elitist replacement policy.
end for
```

Figure 9: Pseudo code of GPSA.

The GPSA starts by generating an initial population $P(0)$ of μ randomly generated points Γ which is composed by α bit-length strings. The inner loop in GPSA incorporates GA operators: recombination, mutation, and replacement. In recombination, a crossover position is randomly selected with a probability of $p_r \in [0,1]$. Each recombined point is mutated with a probability of $p_m \in [0,1]$. Replacement selects the μ_e fittest points ($0 < \mu_e < \mu$) of the generation as the elite set. After N times iterations of the inner loop, the DPS method is applied to the points generated by the evolutionary operators and constructs each sequence of iterates that converge to a stationary point on the mesh parameterized by the step size Δ_{m+1}. The restriction on the replacement strategy ensures that the elite set is kept for further processing.

5 EXPERIMENTAL RESULTS

This section presents a performance comparison of the GPSA and the conventional GA on the well-known 8 benchmark functions whose functional values and coordinates of each global minimum are already reported in (Yao et al., 1999; Schwefel, 1995). Several kinds of benchmark functions are selected to make a generalized conclusion: functions with no local minima (f_1-f_2) and local minima (f_3-f_8). A more detailed description of each function is given in the Appendix.

5.1 Experimental Setup

For both GA and GPSA tests, we used a population size μ of 10 with the elite set $\mu_e = 1$, and for each problem the number of trials was 100. The recombination operator was uniform crossover with a probability $p_c = 0.5$, and the mutation probability was $p_m = 0.001$. The length per object variable in GA was 10, and in GPSA, the initial and final low length per object variable were $\alpha = 3$ and $\beta = 10$, respectively. The number of generations was $4,000$ in GA and $N = 50$ in GA loop of GPSA.

To implement GPSA, it is necessary to determine a proper kind of the generating matrix in digital pattern. We used the standard $2n$ directions, $C = \{e_1, \cdots, e_n, -e_1, \cdots, -e_n\}$, where $e_i \in \mathbf{R}^n$ is the $i-th$ unit vector, because it gives a linear increase of function evaluation with problem dimension.

GA was terminated after $40,000$ function evaluations, and the performance comparisons between GPSA and GA were made based upon the termination point of GPSA. Our experimental analysis considers three performance measures: the number of trials which succeed in attaining to the global optimum for each benchmark function, the number of cost function evaluations during simulations, and the value of the best solution found.

5.2 Numerical Results

Figure 10 shows the performances of GPSA and GA on the benchmark functions. The results of GPSA were selected through 100 independent trials as the best case and the worst case, and the result of GA was averaged over 100 independent trials. The GA converged faster than the GPSA for most functions initially, around 4,000 to 6,000 of function evaluations. However, GPSA overperformed GA obviously while fewer function evaluations. Although GA quickly approaches the neighborhood of the global minimum, GA has a difficulty in obtaining some required accuracy. DPS method's ability to accelerate the search and to refine the solution more evenly helps GA in achieving good performance.

To judge the success of a trial, we used the condition

$$|f^* - \hat{f}| < \varepsilon_1 |f^*| + \varepsilon_2,$$

Table 1: Results of GPSA. The results were averaged over 100 independent trials where "SUCC %" indicates the ratio of trials which succeed in attaining the global optimum , "EVAL #" means the average number of function evaluations, and "VAR" means the variance of trials which succeed. Functions (Yao et al., 1999; Schwefel, 1995): SP (spare function), SC1 (schwefel's problem 2.22), SC2 (schwefel's problem 1.2), SC3 (schwefel's problem 2.26), GR (griewank function), AC (ackley function), RA (rastrign function), and SH (shubert function). And "n" is the number of variables.

Function	SUCC %	EVAL #	VAR
$SP\ (n=10)$	100	5504.35	0.0
$SC1\ (n=10)$	100	4773.82	0.0
$SC2\ (n=10)$	44	5635.07	0.0
$SC3\ (n=2)$	92	2116.56	0.0
$GR\ (n=10)$	100	5321.53	0.0
$AC\ (n=10)$	100	5483.51	0.0
$RA\ (n=10)$	92	7590.75	0.0
$SH\ (n=2)$	38	4240.23	0.0013

where $\hat{f}$ refers to the best function value obtained by GPSA, f^* refers to the known exact global minimum, and ε_1 and ε_2 are small positive numbers. We set ε_1 and ε_2 equal to 10^{-3} and 10^{-6}, respectively. The results are shown in Table 1, where the average number of function evaluations and the variance are related only to successful trials. Table 1 shows that GPSA reached the global minima in a very good success rate for the majority of the tested functions. Moreover, the numbers of function evaluations and the average errors show the efficiency of the method.

On the other hands, GA had few successful trials on any test functions at the termination point of GPSA.

6 CONCLUSIONS

This paper first developed a new class of pattern search method that digitizes the patterns, called the digital pattern search (DPS) method. Then, we presented a new hybrid global search algorithm, the genetic pattern search algorithm (GPSA), which has a self-adapting technique to modify the step size and chase the approximate optimal direction. Applying the DPS method in addition to the ordinary GA operators such as recombination and mutation enhances the exploration process and accelerates the convergence of the proposed algorithm. The experimental results also showed that the GPSA works successfully on some well known test functions.

REFERENCES

Audet, C. and Dennis, Jr., J. E. (2003). Analysis of generalized pattern searches. *SIAM J. on Optim.*, 13(3):889–903.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, MA.

Günal, T. (2000). A hybrid approach to the synthesis of nonuniform lossy transmission-line impedance-matching sections. *Microwave and Optical Technology Letters*, 24:121–125.

Hedar, A. and Fukushima, M. (2004). Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optim. Methods and Software*, 19:291–308.

Horst, R. and Pardalos, P. M. (1995). *Handbook of Global Optimization*. Kluwer Academic Publishers, Boston, MA.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, London, UK.

Musil, M., Wilmut, M. J., and Chapman, N. R. (1999). A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion. *IEEE J. Oceanic Eng.*, 24(3):358–369.

Osman, I. H. and Kelly, J. P. (1996). *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Boston, MA.

Pardalos, P. M. and Romeijn, H. E. (2002). *Handbook of Global Optimization*. Kluwer Academic Publishers, Boston, MA.

Pardalos, P. M., Romeijn, H. E., and Tuy, H. (2000). Recent developments and trends in global optimization. *J. Comput. Appl. Math.*, 124(1-2):209–228.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking: The Sixth Generation*. Addison-Wesley, New York, NY.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM J. on Optim.*, 7(1):1–25.

Yang, R. and Douglas, I. (1998). Simple genetic algorithm with local tuning: efficient global optimizing technique. *J. Optim. Theory Appl.*, 98(2):449–465.

Yao, X., Liu, Y., and Lin, G. (1999). Evolutionary programming made faster. *IEEE Trans. on Evol. Comput.*, 3(2):82–102.

Yen, J., Liao, J., Randolph, D., and Lee, B. (1998). A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Trans. on Syst., Man, and Cybern. B*, 28(2):173–191.

Zentner, R., Sipus, Z., and Bartolic, J. (2001). Optimization synthesis of broadband circularly polarized microstrip antennas by hybrid genetic algorithm. *Microwave and Optical Technology Letters*, 31:197–201.

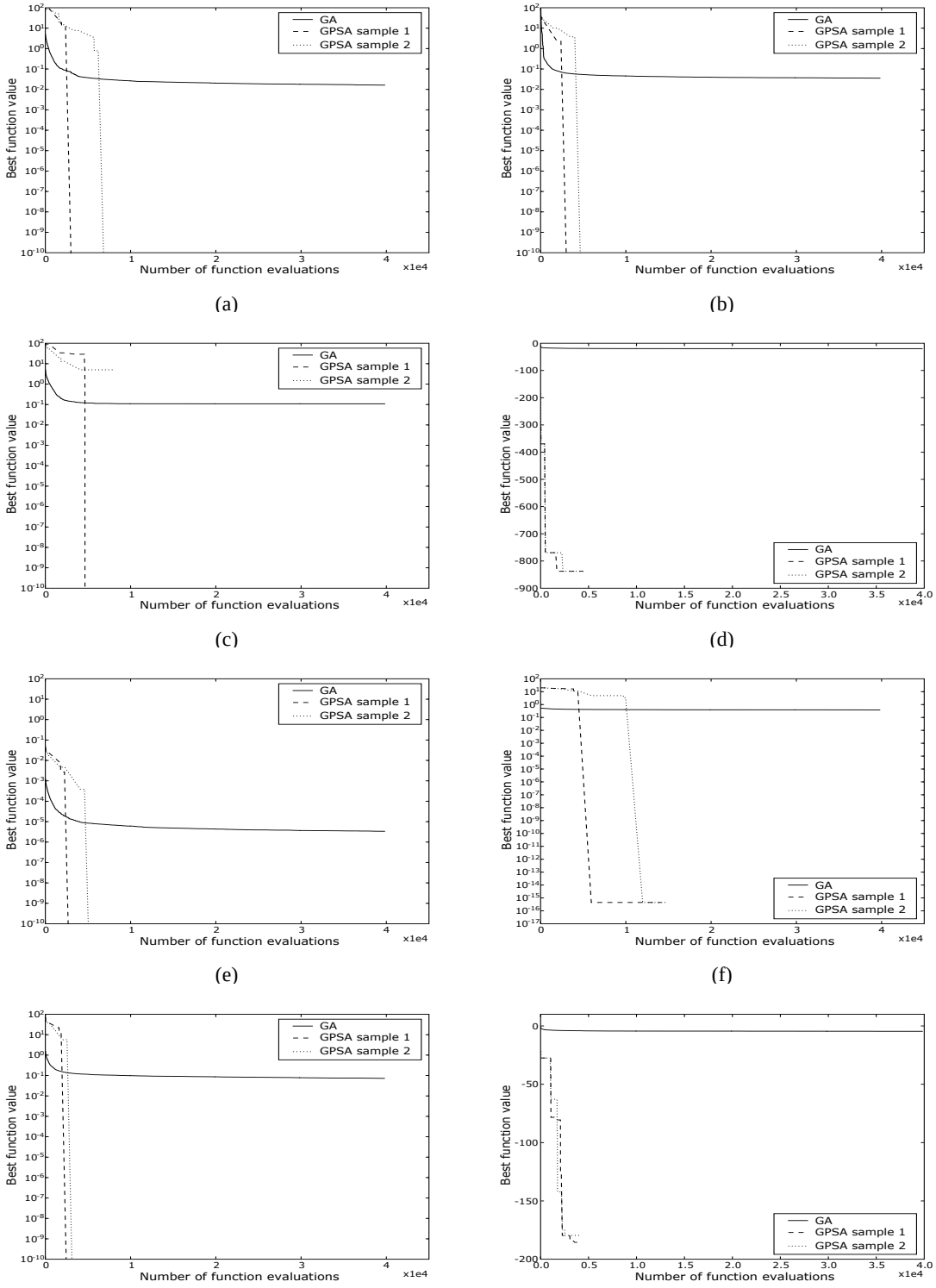

Figure 10: The comparisons of the performance between GPSA and GA. The results of GPSA were selected through 100 independent trials as the best case and the worst case, and the result of GA was averaged over 100 independent trials. (a)-(h) correspond to the results of test functions f_1-f_8, respectively.