Multiple Solution Search Based on Hybridization of Real-Coded Evolutionary Algorithm and Quasi-Newton Method

Satoshi Ono, Member, IEEE, Yusuke Hirotani, and Shigeru Nakayama, Non-Members, IEEE

Abstract—In recent years, many evolutionary computation methods have been proposed and applied to real-world problems; however, gradient methods are regarded as promising for their capacity to solve problems involving real-coded parameters. Addressing real-world problems should not only involve the search for a single optimal solution, but also a set of several quasi-optimal solutions. Although some methods aiming the search for multiple solutions have been proposed (e.g., Genetic Algorithm with Sharing and Immune Algorithm), these could not render highly optimized solutions to real-coded problems. This paper proposes hybrid algorithms combining real-coded evolutionary computation algorithms and gradient search methods for multiple-solution search in multimodal optimization problems. Furthermore, a new evaluation function of solution candidates with gradient is presented and discussed in order to find quasi-optimal solutions. Two hybrid algorithms are proposed—a hybridization between Immune Algorithm and Quasi-Newton method (IA+QN) and a hybridization between Genetic Algorithm with Sharing and Quasi-Newton method (GA+QN). Experimental results have shown that the proposed methods can find optimal and quasi-optimal solutions with high accuracy and efficiency even in high-dimensional multimodal benchmark functions. The results have also shown that GA+QN has better performance and higher robustness in terms of parameter configuration than IA+QN.

I. INTRODUCTION

In real-world optimization problems like architectural structure design, optical lens design or protein structure prediction; availability of several feasible solutions can become more important than a single optimal solution in order to allow designers or engineers to choose the most desirable approach out of the solution candidates. Various algorithms such as Simulated Annealing, Tabu Search, Genetic Algorithm (GA), etc. have been proposed to solve large-scale combinatorial optimization problems in the real world. Nevertheless, these probabilistic search methods are inadequate to find multiple optimal or promising quasi-optimal solutions simultaneously because they were primarily designed to find a single optimal or quasi-optimal solution.

Genetic Algorithm with sharing (GA$S$) [1], Immune Algorithm (IA) [2]–[6] and other iterated genetic algorithms [7] are proposed for simultaneous search of multiple solutions. It is well known that hybridizing evolutionary computation algorithms (which are appropriate for global search) and local search algorithms enhances search performance. Although many studies have been conducted on the hybridization of evolutionary computation algorithms for a single solution and local search algorithms [8]–[11]; as far as we know, little attention has been given to hybridization of search algorithms for multiple solutions and local search algorithms.

In this paper, two methods of simultaneous search for multiple optimal and quasi-optimal solutions are proposed, namely hybridizations between Quasi-Newton (QN) method and real-coded evolutionary computation algorithms (i.e., real-coded IA and GA$S$). These methods also calculate fitness function with gradient, which allows finding multiple solutions with high accuracy and efficiency even in high-dimensional multimodal functions.

Experimental results with up to 15 dimension problems have shown that the proposed methods can find optimal and quasi-optimal solutions with high accuracy and efficiency even in high-dimensional multimodal benchmark functions. The results have also shown that GA$S$+QN has better performance and less dependency on search parameters than IA+QN.

II. PRINCIPLES OF THE PROPOSED METHODS

We propose two hybrid algorithms IA+QN and GA$S$+QN for multiple solution search, based on three principles as follows:

1) Utilizing real-coded genes. In recent years, various genetic operations like crossover and mutation for real-coded GA have been proposed such as linear crossover (LX) [12], blend crossover (BLX-$\alpha$) [13], Simplex Crossover (SPX) [14], and Unimodal Normal Distribution Crossover (UNDX) [15]. Few or no real-coded algorithms, however, have been proposed for multiple solution search. The proposed methods are based on evolutionary algorithms, which aim at simultaneous search for optimal and semi-optimal solutions, and utilize real-coded genes so that solutions of great precision in various real-world problems can be found.

2) Applying Quasi-Newton method to promising solution candidates.

In general, most of niching algorithms work effectively in low dimensional problems [5], [7], [16], [17]. The effectiveness of QN also decreases in higher dimensional problems.

The proposed methods use GA$S$ or IA for well-balanced global search between search intensification and diversification, and QN for careful local search. The hybridization enables multiple solution search even in high dimensional problems on which the
existing niching algorithms and QN can hardly work effectively.

3) Utilizing fitness function calculation method with gradient.

Existing niching methods such as GA, or IA sometimes fail to find quasi-optimal solutions whose values are not so high.

The proposed methods utilize a new evaluation function for multiple solution search based on gradient in order not to miss solutions whose values are not so high. Affinity \( \Phi_v \) of an antibody \( v \) in IA (or fitness \( g(v) \) of an individual \( v \) in GA) is calculated using the following equation:

\[
\Phi_v = \frac{W_1 \times f_{\text{max}}}{1 + |f'(v)| + W_2 \times u_0(f''(v))}.
\]

\( f_{\text{max}} \) is the max value of function \( f \). In case the max value is unknown, a roughly estimated max value is available. Derivatives \( f'(v) \) and \( f''(v) \) are approximately calculated using the following equation:

\[
f'(v) = \frac{f(v + dv) - f(v)}{dv},
\]

\( u_a(t) \) is a step function defined as follows:

\[
u_a(t) = \begin{cases} 1 & t \geq a \\ 0 & t < a \end{cases}.
\]

Figure 1 shows the effects using different parameters \( W_1 \) and \( W_2 \) in equation (1). Substituting 0.0 for \( W_1 \) makes \( \Phi_v = f(v) / f_{\text{max}} \) which just normalized the function equation. Larger \( W_1 \) value makes \( \Phi_v \) of quasi-optimal solutions whose \( f(v) \) is low higher as shown in Figure 1(b), as a result the semi-optimal solutions can be found easily. Using only the first derivative, however, increases \( \Phi_v \) at bottoms of gorges inappropriately as shown Figure 1(b). Substituting sufficiently larger value than 1.0 for \( W_2 \) prevents inappropriate increase of \( \Phi_v \) at the gorge base as shown in Figure 1(c).

III. PROPOSED METHOD I — IA+QN

A. Overview

Inspired by the natural immune system, which is a complex adaptive system employing several mechanisms for defense against foreign pathogens, various optimization and learning methods have been proposed [2], [3], [5], [6], [18]. In this paper, we focus on IA proposed by Mori and Fukuda et al. [2], [3], [6] based on somatic theory and the network hypothesis as general IA. IA is similar to GA except existence of memory cells and suppressor T cells. IA calls a given problem an antigen, and calls a solution candidate antibody. IA has memory cells regarded as solutions found by search. Affinity indicates its effectiveness against the antigen, and corresponds to fitness in GA.

Figure 2 shows the rough algorithms of general GA and the proposed IA+QN; that is, a hybridization of IA and QN. The black-shaded processes are different from general IA [2], [3], [6], [19], [20] — affinity calculation and application of QN. The following explanation is focused on the difference between GA, IA, and the proposed IA+QN.

B. Initial antibodies generation

At first, IA generates \( N \) initial antibodies against a given antigen. An antibody has a chromosome whose length is \( n \), the number of dimensions of the given problem, i.e. the number of design variables, like an individual in GA.

C. Affinity calculation

The proposed IA+QN calculates affinity \( \Phi_v \) of an antibody \( v \) by following equation (1), whereas the affinity equals to \( f(v) \) in general IA.

D. Similarity and density calculation

IA calculates density for each antibody in order to choose advisable solution candidates. In this paper, we define the density \( \Theta_v \) of the antibody \( v \) as the rate of antibodies that have higher similarity to \( v \) than a threshold \( T_{\Psi} \) as shown in the following equation:

\[
\Theta_v = \frac{1}{N} \sum_{i=1}^{N} u_{T_{\Psi}}(\Psi_{vw}).
\]

\( \Psi_{vw} \) represents similarity between antibodies \( v \) and \( w \), and is defined by the following equation:

\[
\Psi_{vw} = \frac{1}{1 + H_{vw}}.
\]

\( H_{vw} \) represents distance between \( v \) and \( w \) calculated by the following equation:

\[
H_{vw} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{v_i-w_i}{D_i^{\text{max}}-D_i^{\text{min}}} \right|.
\]
E. Differentiation of memory cells and suppressor T cells

Whereas general IA differentiates an affinity $v_c$ whose density exceeds a threshold $T_{v_1}$ into memory cell and suppressor T cell, the proposed IA+QN applies QN to $v_c$ before differentiation, and regards $v_c$ applied QN as a memory cell candidate $\mu$.

IA+QN memorizes $\mu$ as a memory cell if one of the following conditions is satisfied:

1) There is no memory cell which has higher similarity to $\mu$ than a threshold $T_{v_1}$, and the number of memory cells is less than the limit $M$.

2) There are more than one memory cells which have higher similarity from $\mu$ than $T_{v_1}$, and the affinity of the most similar memory cell from $\mu$ is lower than that of $\mu$.

3) There is no memory cell which has higher similarity to $\mu$ than a threshold $T_{v_1}$, the number of memory cells reaches the limit $M$, and the affinity of the most similar memory cell from $\mu$ is lower than that of $\mu$.

A suppressor T cell $s$ which has the same chromosome as $\mu$ differentiates into memory cells at the same time when $\mu$ differentiates. $s$ removes antibodies which has the higher similarity to $s$ than a threshold $T_s$. After suppression, new antibodies are generated randomly in order to fill vacancies of $N$ antibodies.

F. Expectation value calculation and antibody generation

In IA and IA+QN, new antibodies are generated as follows. First, half of all antibodies which have higher affinity than the remaining half of all antibodies is selected, and the remaining half is dismissed. Next, $N/4$ parent pairs are selected by tournament selection. IA and IA+QN perform genetic operations with expectation value $E_v$ calculated using the following equation instead of fitness:

$$E_v = \frac{\Phi_v}{\Theta_s} \prod_{u=1}^{S} (1 - \Psi_{v_u} w_{T_{v_2}}(\Psi_{s_u})) \quad (7)$$

where $T_{v_2}$ is a threshold and $S$ is the number of suppressor T cells. And then, $N/2$ children are generated from the selected parents by crossover and mutation.

G. QN Application

IA+QN applies QN to antibodies chosen randomly in addition to antibodies differentiated into memory cells. In each generation, the number of antibodies to which QN is applied is $P_{QN} \times N - N_M$, where $P_{QN}$ is a parameter which represents the rate of antibodies to which QN is applied, and $N_M$ is the number of memory cell candidates in the current generation.

IV. PROPOSED METHOD II — GA$_S$+QN

A. Overview

Figure 2 shows the rough algorithm of the proposed GA$_S$+QN that is a hybridization of GA$_S$ and QN. The black-shaded processes are different from general GA$_S$ — fitness calculation and application of QN.

GA$_S$ uses sharing fitness $g_s$ based on niche number in addition to fitness $g$. GA$_S$+QN selects elite individuals according to $g$, and applies QN to new elite individuals. After elite selection and QN application, GA$_S$+QN calculates $g_s$, and performs genetic operation based on $g_s$, instead of $g$, such as selection for reproduction by roulette selection and sampling with replacement, crossover, mutation, selection for survival by simple GA. Fitness $g(v)$ of an individual $v$ is calculated using equation (1).
The following explanation focuses on the difference between general GA$_5$ and proposed GA$_5$+QN.

B. Elite update

Elite individuals are chosen through the following procedure.

[Step 1] The best $N_e$ individuals are selected as elite group $G_e$ in order of $g$, and remaining $N-N_e$ individuals make group $G_g$.

[Step 2] If there is a pair of two individuals between which distance is lower than the sharing radius $\sigma$, one of the individuals with lower $g$ is moved from $G_e$ to $G_g$, and the best individual in $G_g$ is moved to $G_e$.

[Step 3] If there is no pair whose distance is lower than $\sigma$, then terminate this procedure and let individuals in $G_e$ be elites. Otherwise go back to Step 2.

QN is applied to new elites, an individual that is in $G_e$ and was not in $G_e$ in the last step.

C. Sharing fitness calculation

Sharing fitness $g_s(v)$ of an individual $v$ is calculated using the following equation:

$$g_s(v) = \frac{g(v)}{m_v},$$

where $m_v$ is niche number \[1], \[21] of $v$ defined as following equation:

$$m_v = \sum_{j=1}^{N} s(d(v_i, v_j)).$$

$s(d)$ is a sharing function defined as following equation:

$$s(d) = \max(1 - \frac{d}{\sigma}, 0),$$

where $\sigma$ is sharing radius \[21], \[22]. Distance $d(v_i, v_j)$ between individuals $v_i$ and $v_j$ is calculated by equation \[6]. Genetic operators such as selection and crossover are performed based on $g_s$.

D. QN Application

GA$_5$+QN applies QN to individuals chosen randomly in addition to new elites like IA$_5$+QN. In each generation, the number of individuals in $G_e$ to which QN are applied is $\{P_{QN} \times N - N_M\}$, where $N_M$ is the number of new elites in the current generation.

V. EVALUATION

To see the effect of the proposed hybridization and affinity (or fitness) calculation with gradient in multiple solution search, we performed experiments with four test functions.

A. Test functions

Following three multimodal functions are widely used as benchmark problems of multiple solution search \[2], \[3], \[5], \[7], \[17], \[22].

$$F1(x) = \prod_{k=1}^{n} \sin^6(a_k \pi x_k)$$

$$F2(x) = \prod_{k=1}^{n} e^{-((\log_2(2^{k-1/2})/a_k))^2} \sin^6(a_k \pi x_k)$$

$F1$ is a function involving plural optimal solutions, and $F2$ is a function involving one optimal and plural semi-optimal solutions, with $a_1 = 4$, $a_2 = 2$, and $a_3 = 3$.

$$F3(x) = -\sum_{j=1}^{5} \sum_{i=1}^{5} i \cos((i + 1)x_j + i)$$
$F_3$ is a Shubert function involving 18 optimal solutions and 742 semi-optimal solutions.

Although various functions like Rastrigin or Rosenbrock are used as benchmarks [23], there are no or few high-dimensional multimodal functions involving more than 10 optimal solutions appropriate for benchmarks of multiple solution search. We, therefore, define a benchmark function $F_4$ by combining $F_1$ and $F_2$ for each dimension in order to evaluate the multiple solution search performance of the proposed methods in high dimensional functions like $n = 5$, 10, and 15.

$$F_4(x) = \prod_{k=1}^{n} F'_4(x)$$

$$F'_4(x) = \begin{cases} \sin^6(a_k \pi x_k) & \text{if } n = 5 \land k = 5 \\ (\text{or } n = 10 \land (k = 2 \lor k = 10)) & \\ \sin^6(a_k \pi x_k) & \text{(otherwise)} \end{cases}$$

$$a_k = 1 + \left( (k - 1) \mod 5 \right)$$

Figure 3 shows $F_1$, $F_2$, and $F_3$ in two dimensions. Table I shows the numbers of optimal and semi-optimal solutions. The purpose in $F_1$, $F_3$, and $F_4$ is to find all optimal solutions, and that in $F_2$ is to find all optimal and all quasi-optimal solutions. The number of other local solutions, which are not objective solutions of search in $F_3$ and $F_4$, is also shown in Table I. We assume a sufficient condition of finding a solution to be that a distance between a solution candidate and an optimal (or quasi-optimal) solution is lower than $1.0 \times 10^{-8}$.

**B. Compared algorithms and parameter configuration**

IA, GA$_S$, a hybridization algorithm of Island GA and QN (IGA+QN) [10], and an iterative QN with random initialization (IQN) are used for comparison with proposed IA+QN and GA$_S$+QN. All methods use real-coded variables.

IQN has the same memory as IA and IA+QN as we noted in III-E. QN does not use equation (1) in its algorithm, and IQN uses equation (1) only after QN to judge whether the solution candidate should be remembered or not. IGA+QN does not perform migration operation in order to maintain search diversity.

Parameters were configured as shown in Table II. The number of affinities or individuals was set to 250 or 500. IA, GA$_S$, IGA+QN, IA+QN, and GA$_S$+QN used BLX-$\alpha$ as a crossover operator with $\alpha = 0.5$. The number of elites in GA$_S$ and GA$_S$+QN $N_e$, islands in IGA+QN, and memory cells in IA and IA+QN were set to 25. To preserve 25 elites, the crossover rate $P_{cross}$ was set to 0.9 in the case that the number of individuals equals to 250, or 0.95 in the case that the number of individuals equals to 250 in GA, GA$_S$, and GA$_S$+QN. Mutation rate $P_{mutate}$ was set to 0.1.

Values of IA parameters such as $T_{II}$ and $T_{e1}$ were determined according to the previous works [2], [3], [6] and preliminary experiments [24] using $F_1$, $F_2$, and $F_3$ in $n \leq 2$. Sharing radius $\rho$ was set to the same value as $d$ in IA and IA+QN.

30 runs were performed for each methods and each run continues until all solutions are found or the function calls reaches $(n-1) \times 10^7$ when $n < 5$ or $\lfloor n/5 \rfloor \times 10^7$ when $n \geq 5$. An evaluation of a solution candidate with equation (1) requires three function calls when $W_1 > 0$.

We evaluate the algorithms by measuring their $S_{All} [%]$ (the rate of runs in which the algorithm succeeds in finding all solutions), $F_{Opt} [%]$ (mean discover rate of optimal solutions), $F_{Semi} [%]$ (mean discover rate of quasi-optimal solutions), $E_{Opt}$ (mean number of function calls until finding an optimal solution), and $E_{All}$ (mean function calls until find all optimal and semi-optimal solutions).

We tested an application rate of QN $P_{QN}$ with 40 [%], 80 [%], and ‘Elite Only’ (EO), which allows IA+QN and GA$_S$+QN to apply QN to only promising solution candidates: memory cell candidates in IA+QN and new elite individuals in GA$_S$+QN. In case of $P_{QN} = EO$, if there is no promising candidates in a generation, an antibody (or an individual) is selected randomly and QN is applied to the antibody.

The proposed affinity calculation equation (1) involves two weight parameters $W_1$ and $W_2$. $W_2$ does not require detailed adjustment: much higher value than 1.0 like 1.0 $\times 10^2$ or 1.0 $\times 10^5$ is sufficient for $W_2$ to work properly as shown in Figure 1(c) and (e). We therefore focus on the effectiveness of $W_1$ varying with 0.0, 1.0, and 2.0.

**C. Experimental results in $F_1$, $F_2$, and $F_3$**

Table III shows results of IQN, IA, GA$_S$, IGA+QN, IA+QN, and GA$_S$+QN in functions $F_1$ ($n = 3$), $F_2$($n = 3$), and $F_3$($n = 2$). A brief observation in Table III shows that IQN, IA+QN, and GA$_S$+QN could find all solutions simultaneously, and IA, GA$_S$, and IGA+QN could not.

IA and GA$_S$ could find all optimal solutions with the accuracy of about 1.0 $\times 10^{-4}$, but could not find the solutions with the accuracy of 1.0 $\times 10^{-8}$. IGA+QN succeeded in finding more than half of the solutions, but could not find all of them.

IQA could find all solutions with 100% accuracy and $E_{All}$ was lower than IA+QN and GA$_S$+QN. IA+QN and GA$_S$+QN also succeeded in finding all solutions with 100% accuracy. In $F_3$, IA+QN found all solutions faster than IQN in the case $N = 250$, $W_1 = 0$, and $P_{QN} = EO$ in particular. In $F_1$, GA$_S$+QN found all solutions faster than IQN in the case $N = 250$, $W_1 = 0$, and $P_{QN} = EO$. In addition, results in $F_2$ suggests that GA$_S$+QN was more robust against parameter configuration than IA+QN.

**D. Experimental results in $F_4$**

Table IV shows results of the algorithms that showed good search performance in $F_1$, $F_2$, and $F_3$: IQN, IA+QN, and GA$_S$+QN, in function $F_4$. A brief observation in Table IV
<table>
<thead>
<tr>
<th>Method</th>
<th>N</th>
<th>W</th>
<th>S</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>All</td>
<td>Opt</td>
<td>Opt</td>
<td>Opt</td>
<td>All</td>
</tr>
<tr>
<td>[Previous methods]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQN</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>IA</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>GA</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>14.60×10⁸</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>6.60×10⁶</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>9.40×10⁶</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RGA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>43.87×10⁸</td>
<td>0</td>
<td>57</td>
<td>41.2×10⁸</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>48.11×10⁴</td>
<td>0</td>
<td>50</td>
<td>45.1×10⁴</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>49.11×10⁴</td>
<td>0</td>
<td>60</td>
<td>48.31×10⁴</td>
<td></td>
</tr>
<tr>
<td>[Proposed methods]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>5.5×10⁸</td>
<td>9.5×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>9.4×10⁸</td>
<td>4.1×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.2×10⁸</td>
<td>4.4×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>3.1×10⁸</td>
<td>7.5×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.1×10⁸</td>
<td>4.0×10⁸</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.3×10⁸</td>
<td>5.0×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>3.4×10⁸</td>
<td>8.4×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.0×10⁸</td>
<td>4.0×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.2×10⁸</td>
<td>4.3×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>GA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>1.3×10⁸</td>
<td>7.3×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.3×10⁸</td>
<td>5.7×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.6×10⁸</td>
<td>5.6×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>6.7×10⁸</td>
<td>7.5×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.4×10⁸</td>
<td>4.7×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.8×10⁸</td>
<td>4.8×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>6.9×10⁸</td>
<td>8.1×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.3×10⁸</td>
<td>4.7×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.7×10⁸</td>
<td>4.4×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>GA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>1.6×10⁸</td>
<td>5.5×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>5.6×10⁸</td>
<td>1.3×10⁹</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>8.9×10⁸</td>
<td>1.6×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>1.9×10⁸</td>
<td>8.0×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>5.4×10⁸</td>
<td>1.3×10⁹</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>9.7×10⁸</td>
<td>1.6×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>3.8×10⁸</td>
<td>1.2×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>5.7×10⁸</td>
<td>1.4×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>9.7×10⁸</td>
<td>1.6×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>GA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>1.8×10⁸</td>
<td>6.6×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.1×10⁸</td>
<td>1.7×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>1.9×10⁸</td>
<td>2.0×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>IA+QN</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>2.2×10⁸</td>
<td>1.2×10⁸</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0</td>
<td>100</td>
<td>1.1×10⁸</td>
<td>1.6×10⁸</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0</td>
<td>100</td>
<td>2.0×10⁸</td>
<td>2.1×10⁸</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

 shows that IQN could not find all solutions in the case \( n \geq 10 \), and IA+QN and GA+QN could even in the case \( n = 15 \).

QN could find all solutions in F4 with \( n = 5 \), but numerous local optimal solutions disturbed QN in the case \( n \geq 10 \), resulting QN failed to find even one optimal solution in the case \( n = 15 \).

Although both IA+QN and GA+QN could find all optimal solutions even when \( n = 15 \), it is obvious that
GA$_S$+QN was more robust against parameter configuration than IA+QN. In F4 with $n = 15$, GA$_S$+QN succeeded in finding all solutions when mainly $P_{QN} = 40$, $W_1 = 0$, or $N = 250$, whereas IA+QN succeeded only when $P_{QN} = EO$ and $W_1 = 0$. Focusing on $E_{All}$ reveals that IA+QN with $P_{QN} = EO$ and $W_1 = 0$ was almost the fastest of the three algorithms.

VI. DISCUSSION

A. Parameter $W_1$

Parameter $W_1$ influences the performance of multiple solution search. From Table III and IV, all tested algorithms worked better when $W_1 = 0.0$ than when $W_1 > 0.0$ in problems whose objective is to find plural optimal solutions such as F1, F3 and F4. $W_1$ higher than 0.0 increased affinity (or fitness) of local optimal solutions and reduce areas of hillsides as shown in Figure 1; consequently the search performance was aggravated.

In contrast to F1, F3, and F4, $W_1$ set to 1.0 or 2.0 helped the tested algorithms find quasi-optimal solutions in F2 whose objective is to find optimal solution and all quasi-optimal solutions. $W_1$ should therefore be determined depending on the purpose — If quasi-optimal solutions are necessary, $W_1$ should be set 1.0 or higher.

B. Parameter $P_{QN}$

Parameter $P_{QN}$ influences search cost balance between IA (or GA$_S$) and QN. In problems whose objective is to find multiple optimal solutions such as F1, F3 and F4, proposed IA+QN and GA$_S$+QN found all solutions quickly when $P_{QN} = EO$ in which QN was applied to only promising solution candidates. In F2 in which quasi-optimal solutions must be founded, IA+QN and GA$_S$+QN succeeded in finding all quasi-optimal solutions when $P_{QN} = 40$; this shows that applying QN to solution candidates chosen randomly aids in finding quasi-optimal solutions. GA$_S$+QN with lower $P_{QN}$
could find all solutions more quickly than with higher $P_{O N}$, because evolution of $G_{A_3}$ can search for solutions better, without fall into local optima, than $Q N$ in high dimensional problems like $n \geq 10$.

C. Comparison between IA+QN and $G_{A_3}+QN$

One difference between behaviors of IA and $G_{A_3}$ is whether density of solution candidates is considered explicitly (in IA) or implicitly (in $G_{A_3}$); IA obtains promising candidates from gathered antibodies, whereas $G_{A_3}$ simply regards individuals with high fitness as the candidates. Other difference between them is whether promising solution candidates are used for reproduction (in $G_{A_3}$) or not (in IA); elite individuals are subjects of crossover and mutation, whereas memory cells are not. These two differences yield a little difference of search performance between IA and $G_{A_3}$ in functions $F_1$, $F_2$, and $F_3$.

Combining QN emphasizes the former difference. Applying QN to an antibody (or an individual) chosen randomly may improve the antibody to be a promising candidate. IA+QN does not admit an isolated antibody as a memory cell even though the antibody has the highest affinity, whereas $G_{A_3}+QN$ treats an isolated promising individual as an elite immediately. This causes the difference of robustness against parameter configuration of $P_{O N}$ and $W_1$ between IA+QN and $G_{A_3}+QN$ in $F_2$ and $F_4$ with $n \geq 10$.

VII. CONCLUSION

In this paper, we proposed hybrid algorithms of evolutionary computation and gradient search, and evaluation function for a solution candidate with gradient of fitness landscape. We performed experiments to compare proposed IA+QN and $G_{A_3}+QN$ with other conventional algorithms in high dimensional functions. Experimental results showed how the hybridization enables to find all solutions in high dimension functions. In addition, the results showed the evaluation function of solution candidates with gradient helps all tested algorithms find quasi-optimal solutions.

In the future, we plan to apply the proposed methods to real-world problems such as building structure design and optical lens design.

REFERENCES


