# Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization

Kuk-Hyun Han and Jong-Hwan Kim

***Abstract*—This paper proposes a novel evolutionary algorithm inspired by quantum computing, called a quantum-inspired evolutionary algorithm (QEA), which is based on the concept and principles of quantum computing, such as a quantum bit and superposition of states. Like other evolutionary algorithms, QEA is also characterized by the representation of the individual, the evaluation function, and the population dynamics. However, instead of binary, numeric, or symbolic representation, QEA uses a Q-bit, defined as the smallest unit of information, for the probabilistic representation and a Q-bit individual as a string of Q-bits. A Q-gate is introduced as a variation operator to drive the individuals toward better solutions. To demonstrate its effectiveness and applicability, experiments are carried out on the knapsack problem, which is a well-known combinatorial optimization problem. The results show that QEA performs well, even with a small population, without premature convergence as compared to the conventional genetic algorithm.**

***Index Terms*—Evolutionary algorithm, knapsack problem, probabilistic representation, quantum computing.**

## I. Introduction

EVOLUTIONARY algorithms (EAs) are principally a stochastic search and optimization method based on the principles of natural biological evolution. Compared to traditional optimization methods, such as calculus-based and enumerative strategies, EAs are robust, global, and may be applied generally without recourse to domain-specific heuristics, although performance is affected by these heuristics. The three main-stream methods of evolutionary computation which have been established over the past 45 years are genetic algorithms (GAs) developed by Fraser [1], Bremermann [2], and Holland [3], evolutionary programming (EP) developed by Fogel [4], and evolution strategies (ES) developed by Rechenberg [5] and Schwefel [6].

EAs operate on a population of potential solutions, applying the principle of survival of the fittest to produce successively better approximations to a solution. At each generation of the EA, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and reproducing them using variation operators. This process may lead to the evolution of populations of individuals that are better suited to their environment than the individuals from which they were created, just as in natural adaptation.

Manuscript received April 11, 2000; revised April 5, 2002 and June 24, 2002.

The authors are with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea (e-mail: khhan@vivaldi.kaist.ac.kr; johkim@vivaldi.kaist.ac.kr).

Digital Object Identifier 10.1109/TEVC.2002.804320

EAs are characterized by the representation of the individual, the evaluation function representing the fitness level of the individuals, and the population dynamics such as population size, variation operators, parent selection, reproduction and inheritance, survival competition method, etc. To have a good balance between exploration and exploitation, those components should be designed properly. In particular, in this paper the representation and variation operators are investigated to represent the individuals effectively to explore the search space with the small number of individuals (even with only one individual for real-time application) and to exploit the global solution in the search space within a short span of time, respectively. For these purposes, some concepts of quantum computing are adopted in the proposed evolutionary algorithm.

Quantum mechanical computers were proposed in the early 1980s [7] and the description of quantum mechanical computers was formalized in the late 1980s [8]. Many efforts on quantum computers have progressed actively since the early 1990s because these computers were shown to be more powerful than classical computers on various specialized problems. There are well-known quantum algorithms such as Shor's quantum factoring algorithm [9], [10] and Grover's database search algorithm [11], [12]. Research on merging evolutionary computing and quantum computing has been started since late 1990s. They can be classified into two fields. One concentrates on generating new quantum algorithms using automatic programming techniques such as genetic programming [13]. The other concentrates on quantum-inspired evolutionary computing for a classical computer, a branch of study on evolutionary computing that is characterized by certain principles of quantum mechanics such as standing waves, interference, coherence, etc. In [14] and [15], the concept of interference was included in a crossover operator.

This paper proposes a novel evolutionary algorithm, called a quantum-inspired evolutionary algorithm (QEA), which is based on the concept and principles of quantum computing such as a quantum bit and superposition of states. Like the EAs, QEA is also characterized by the representation of the individual, the evaluation function, and the population dynamics. However, instead of binary, numeric, or symbolic representation, QEA uses a Q-bit as a probabilistic representation, defined as the smallest unit of information. A Q-bit individual is defined by a string of Q-bits. The Q-bit individual has the advantage that it can represent a linear superposition of states (binary solutions) in search space probabilistically. Thus, the Q-bit representation has a better characteristic of population diversity than other representations. A Q-gate is also defined as a variation operator of QEA to drive the individuals toward better solutions and eventually toward a single state. Initially, QEA can represent diverse individuals probabilistically because a Q-bit individual represents the linear superposition of all possible states with the same probability. As the probability of each Q-bit approaches either 1 or 0 by the Q-gate, the Q-bit individual converges to a single state and the diversity property disappears gradually. By this inherent mechanism, QEA can treat the balance between exploration and exploitation. It should be noted that although QEA is based on the concept of quantum computing, QEA is not a quantum algorithm, but a novel evolutionary algorithm for a classical computer [16], [17]. To demonstrate its performance, experiments are carried out on the knapsack problem. The results show that QEA performs well—even with a small population—without premature convergence as compared to the conventional genetic algorithm.

This paper is organized as follows. Section II describes QEA. Section III presents an application example with QEAs and conventional GAs for knapsack problems, and summarizes the experimental results. Section IV analyzes the characteristics of QEA. Concluding remarks follow in Section V.

## II. QEA

Before describing QEA, the basics of quantum computing are addressed briefly in the following. The smallest unit of information stored in a two-state quantum computer is called a quantum bit or qubit [18]. A qubit may be in the "1" state, in the "0" state, or in any superposition of the two. The state of a qubit can be represented as

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where $\alpha$ and $\beta$ are complex numbers that specify the probability amplitudes of the corresponding states. $|\alpha|^2$ gives the probability that the qubit will be found in the "0" state and $|\beta|^2$ gives the probability that the qubit will be found in the "1" state. Normalization of the state to unity guarantees

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2}$$

The state of a qubit can be changed by the operation with a quantum gate. A quantum gate is a reversible gate and can be represented as a unitary operator $U$ acting on the qubit basis states satisfying $U^\dagger U = UU^\dagger$, where $U^\dagger$ is the hermitian adjoint of $U$. There are several quantum gates, such as the NOT gate, controlled NOT gate, rotation gate, Hadamard gate, etc. [18]. If there is a system of $m$ qubits, the system can represent $2^m$ states at the same time. However, in the act of observing a quantum state, it collapses to a single state.

Inspired by the concept of quantum computing, QEA is designed with a novel Q-bit representation, a Q-gate as a variation operator, and an observation process. We present the representation and the proposed algorithm in the following.

### A. Representation

A number of different representations can be used to encode the solutions onto individuals in evolutionary computation. The representations can be classified broadly as: binary, numeric, and symbolic [19]. QEA uses a new representation, called a Q-bit, for the probabilistic representation that is based on the concept of qubits, and a Q-bit individual as a string of Q-bits, which are defined below.

*Definition 1:* A *Q-bit* is defined as the smallest unit of information in QEA, which is defined with a pair of numbers $(\alpha, \beta)$ as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

where $|\alpha|^2 + |\beta|^2 = 1$. $|\alpha|^2$ gives the probability that the Q-bit will be found in the "0" state and $|\beta|^2$ gives the probability that the Q-bit will be found in the "1" state.

A Q-bit may be in the "1" state, in the "0" state, or in a linear superposition of the two.

*Definition 2:* A *Q-bit individual* as a string of $m$ Q-bits is defined as

$$\left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{array}\right] \tag{3}$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1$, $i = 1, 2, \ldots, m$.

Q-bit representation has the advantage that it is able to represent a linear superposition of states. If there is, for instance, a three-Q-bit system with three pairs of amplitudes such as

$$\left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array}\right] \tag{4}$$

then the states of the system can be represented as

$$\frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle - \tfrac{1}{4}|010\rangle - \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle - \frac{1}{4}|110\rangle - \frac{\sqrt{3}}{4}|111\rangle. \tag{5}$$

The above result means that the probabilities to represent the states $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, and $|111\rangle$ are 1/16, 3/16, 1/16, 3/16, 1/16, 3/16, 1/16, and 3/16, respectively. By consequence, the three-Q-bit system of (4) contains the information of eight states.

Evolutionary computing with Q-bit representation has a better characteristic of population diversity than other representations, since it can represent linear superposition of states probabilistically. Only one Q-bit individual such as (4) is enough to represent eight states, but in binary representation at least eight strings, (000), (001), (010), (011), (100), (101), (110), and (111) are needed.

### B. QEA

The structure of QEA is described in the following.

```
Procedure QEA
begin
        $t \leftarrow 0$
i)      initialize $Q(t)$
ii)     make $P(t)$ by observing the states of $Q(t)$
iii)    evaluate $P(t)$
iv)     store the best solutions among $P(t)$ into
        $B(t)$
        while (not termination-condition) do
        begin
```

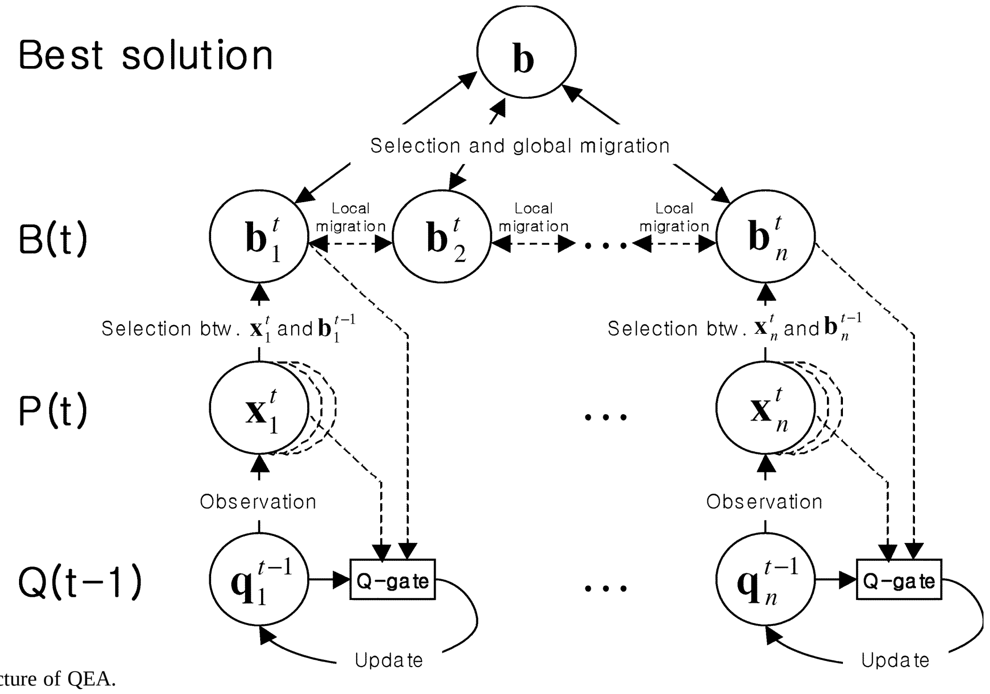


Fig. 1. Overall structure of QEA.

```
            $t \leftarrow t+1$
v)          make $P(t)$ by observing the states of
            $Q(t-1)$
vi)         evaluate $P(t)$
vii)        update $Q(t)$ using Q-gates
viii)       store the best solutions among $B(t-1)$
            and $P(t)$ into $B(t)$
ix)         store the best solution b among $B(t)$
x)          if (migration-condition)
            then migrate b or $b_j^t$ to $B(t)$ globally
            or locally, respectively
        end
end
```

QEA is a probabilistic algorithm similar to other evolutionary algorithms. QEA, however, maintains a population of Q-bit individuals, $Q(t) = \{\mathbf{q}_1^t, \mathbf{q}_2^t, \ldots, \mathbf{q}_n^t\}$ at generation $t$, where $n$ is the size of population, and $\mathbf{q}_j^t$ is a Q-bit individual defined as

$$\mathbf{q}_j^t = \left[\begin{array}{c|c|c|c} \alpha_{j1}^t & \alpha_{j2}^t & \cdots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \cdots & \beta_{jm}^t \end{array}\right] \tag{6}$$

where $m$ is the number of Q-bits, i.e., the string length of the Q-bit individual, and $j = 1, 2, \ldots, n$. Fig. 1 shows the overall structure of QEA. The procedure of QEA is described as follows.

i) In the step of "initialize $Q(t)$," $\alpha_i^0$ and $\beta_i^0$, $i = 1, 2, \ldots, m$, of all $\mathbf{q}_j^0 = \mathbf{q}_j^t|_{t=0}$, $j = 1, 2, \ldots, n$, are initialized with $1/\sqrt{2}$. It means that one Q-bit individual, $\mathbf{q}_j^0$ represents the linear superposition of all possible states with the same probability

$$\left|\Psi_{\mathbf{q}_j^0}\right\rangle = \sum_{k=1}^{2^m} \frac{1}{\sqrt{2^m}} \left|X_k\right\rangle \tag{7}$$

where $X_k$ is the $k$th state represented by the binary string $(x_1 x_2 \cdots x_m)$, where $x_i$, $i = 1, 2, \ldots, m$, is either 0 or 1 according to the probability of either $|\alpha_i^0|^2$ or $|\beta_i^0|^2$, respectively.

ii) This step makes binary solutions in $P(0)$ by observing the states of $Q(0)$, where $P(0) = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots, \mathbf{x}_n^0\}$ at generation $t = 0$. One binary solution, $\mathbf{x}_j^0$, $j = 1, 2, \ldots, n$, is a binary string of length $m$, which is formed by selecting either 0 or 1 for each bit using the probability, either $|\alpha_i^0|^2$ or $|\beta_i^0|^2$, $i = 1, 2, \ldots, m$, of $\mathbf{q}_j^0$, respectively. In a quantum computer, in the act of observing a quantum state, it collapses to a single state. However, collapsing into a single state does not occur in QEA, since QEA is working on a classical computer, not a quantum computer.

iii) Each binary solution $\mathbf{x}_j^0$ is evaluated to give a level of its fitness.

iv) The initial best solutions are then selected among the binary solutions, $P(0)$, and stored into $B(0)$, where $B(0) = \{\mathbf{b}_1^0, \mathbf{b}_2^0, \ldots, \mathbf{b}_n^0\}$, and $\mathbf{b}_j^0 (=\mathbf{b}_j^t|_{t=0})$ is the same as $\mathbf{x}_j^0$ at the initial generation.

v, vi) In the **while** loop, binary solutions in $P(t)$ are formed by observing the states of $Q(t-1)$ as in step ii), and each binary solution is evaluated for the fitness value. It should be noted that $\mathbf{x}_j^t$ in $P(t)$ can be formed by multiple observations of $\mathbf{q}_j^{t-1}$ in $Q(t-1)$. In this case, $\mathbf{x}_j^t$ should be replaced by $\mathbf{x}_{jl}^t$, where $l$ is an observation index.

vii) In this step, Q-bit individuals in $Q(t)$ are updated by applying Q-gates defined below.

*Definition 3:* A *Q-gate* is defined as a variation operator of QEA, by which operation the updated Q-bit should satisfy the normalization condition, $|\alpha'|^2 + |\beta'|^2 = 1$, where $\alpha'$ and $\beta'$ are the values of the updated Q-bit.

The following rotation gate is used as a Q-gate in QEA, such as

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \tag{8}$$

where $\Delta\theta_i$, $i = 1, 2, \ldots, m$, is a rotation angle of each Q-bit toward either 0 or 1 state depending on its sign. $\Delta\theta_i$ should be designed in compliance with the application problem. In the knapsack problem, presented in Section III-A, $\Delta\theta_i$ is obtained as a function of the $i$th bit of the best solution $\mathbf{b}_j^t$ and the $i$th bit of the binary solution $\mathbf{x}_j^t$. It should be noted that NOT gate, controlled NOT gate, or Hadamard gate can be used as a Q-gate. NOT gate changes the probability of the 1 (or 0) state to that of the 0 (or 1) state. It can be used to escape a local optimum. In controlled NOT gate, one of the two bits should be a control bit. If the control bit is 1, the NOT operation is applied to the other bit. It can be used for the problems which have a large dependency of two bits. The Hadamard gate is suitable for the algorithms which use the phase information of Q-bit, as well as the amplitude information.

viii, ix) The best solutions among $B(t-1)$ and $P(t)$ are selected and stored into $B(t)$, and if the best solution stored in $B(t)$ is fitter than the stored best solution $\mathbf{b}$, the stored solution $\mathbf{b}$ is replaced by the new one.

x) If a migration condition is satisfied, the best solution $\mathbf{b}$ is migrated to $B(t)$, or the best one among some of the solutions in $B(t)$ is migrated to them. The migration condition is a design parameter, and the migration process defined below can induce a variation of the probabilities of a Q-bit individual.

*Definition 4:* A *migration* in QEA is defined as the process of copying $\mathbf{b}_j^t$ in $B(t)$ or $\mathbf{b}$ to $B(t)$. A *global migration* is implemented by replacing all the solutions in $B(t)$ by $\mathbf{b}$, and a *local migration* is implemented by replacing some of the solutions in $B(t)$ by the best one of them.

The binary solutions in $P(t)$ are discarded at the end of the loop because $P(t+1)$ will be produced by observing the updated $Q(t)$ in step vii). Until the termination condition is satisfied, QEA is running in the **while** loop.

## III. Application Example

In this section, we present the detailed algorithm of QEA for the knapsack problem. The knapsack problem is considered to demonstrate the applicability of QEA to the combinatorial optimization problem. For the purpose of comparison, three types of GA method are described briefly. The knapsack problem can be described as selecting from among various items those items that are most profitable, given that the knapsack has limited capacity. The 0–1 knapsack problem is described as follows. Given a set of $m$ items and a knapsack, select a subset of the items to maximize the profit $f(\mathbf{x})$

$$f(\mathbf{x}) = \sum_{i=1}^{m} p_i x_i$$

subject to

$$\sum_{i=1}^{m} w_i x_i \leq C$$

where $\mathbf{x} = (x_1 \cdots x_m)$, $x_i$ is 0 or 1, $p_i$ is the profit of item $i$, $w_i$ is the weight of item $i$, and $C$ is the capacity of the knapsack. If $x_i = 1$, the $i$th item is selected for the knapsack.

### A. QEA for the Knapsack Problem

QEA for the knapsack problem consists of a basic structure of QEA and a repair process to satisfy the capacity constraint. The algorithm can be written as follows.

```
Procedure QEA for the Knapsack Problem
begin
    $t \leftarrow 0$
    initialize $Q(t)$
    make $P(t)$ by observing the states of
      $Q(t)$
    repair $P(t)$
    evaluate $P(t)$
    store the best solutions among $P(t)$
      into $B(t)$
    while ($t < \mathrm{MAX\_GEN}$) do
    begin
      $t \leftarrow t+1$
      make $P(t)$ by observing the states of
        $Q(t-1)$
      repair $P(t)$
      evaluate $P(t)$
      update $Q(t)$
      store the best solutions among $B(t-1)$
        and $P(t)$ into $B(t)$
      store the best solution $\mathbf{b}$ among $B(t)$
      if (migration-period)
      then migrate $\mathbf{b}$ or $\mathbf{b}_j^t$ to $B(t)$ globally
        or locally, respectively
    end
end
```

A Q-bit individual of length $m$ represents a linear superposition of solutions to the problem. The length of the Q-bit individual is the same as the number of items. The initialization step is the same as that of the basic structure of QEA in Section II-B. The $i$th item can be selected for the knapsack with a probability of $|\beta_i|^2$ or $(1-|\alpha_i|^2)$. For every bit in the binary string $\mathbf{x}_j^t$, $j = 1, 2, \ldots, n$, in $P(t)$, a random number $r$ is generated from the range $[0 \cdots 1]$; if $r < |\beta_i|^2$, the bit of the binary string is set to 1. Thus, a binary string of length $m$ is formed from the Q-bit individual, which represents a solution observed from the $j$th Q-bit individual. For notational simplicity, $\mathbf{x}$ and $\mathbf{q}$ are used instead of $\mathbf{x}_j^t$ and $\mathbf{q}_j^t$, respectively. To obtain the binary string $\mathbf{x}$, the step of "**make** $P(t)$ by observing the states of $Q(t)$" can be implemented for each Q-bit individual as follows.

```
Procedure Make ($\mathbf{x}$)
begin
    $i \leftarrow 0$
    while ($i < m$) do
    begin
      $i \leftarrow i+1$
      if $random[0, 1) < |\beta_i|^2$
      then $x_i \leftarrow 1$
      else $x_i \leftarrow 0$
    end
end
```

When the binary string violates the capacity constraint, the following repair algorithm is employed.

```
Procedure Repair ($\mathbf{x}$)
begin
  knapsack-overfilled ← false
  if $\sum_{i=1}^{m} w_i x_i > C$
  then knapsack-overfilled ← true
  while (knapsack-overfilled) do
  begin
    select an $i$th item from the knapsack
    $x_i \leftarrow 0$
    if $\sum_{i=1}^{m} w_i x_i \le C$
    then knapsack-overfilled ← false
  end
  while (not knapsack-overfilled) do
  begin
    select a $j$th item from the knapsack
    $x_j \leftarrow 1$
    if $\sum_{i=1}^{m} w_i x_i > C$
    then knapsack-overfilled ← true
  end
  $x_j \leftarrow 0$
end
```

The **update** procedure of Q-bits is presented in the following.

```
Procedure Update ($\mathbf{q}$)
begin
  $i \leftarrow 0$
  while ($i < m$) do
  begin
    $i \leftarrow i + 1$
    determine $\Delta\theta_i$ with the lookup table
    obtain $(\alpha_i', \beta_i')$ from the following:
      if ($\mathbf{q}$ is located in the first/third
        quadrant)
      then $[\alpha_i' \, \beta_i']^T = U(\Delta\theta_i)[\alpha_i \, \beta_i]^T$
      else $[\alpha_i' \, \beta_i']^T = U(-\Delta\theta_i)[\alpha_i \, \beta_i]^T$
  end
  $\mathbf{q} \leftarrow \mathbf{q}'$
end
```

A rotation gate $U(\Delta\theta_i)$ is employed to update a Q-bit individual $\mathbf{q}$ as a variation operator. $(\alpha_i, \beta_i)$ of the $i$th Q-bit is updated as follows:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \qquad (9)$$

Fig. 2 depicts the polar plot of the rotation gate for Q-bit individuals. In this knapsack problem, the angle parameters used for the rotation gate are shown in Table I. Let us define an angle vector $\Theta = [\theta_1 \, \theta_2 \cdots \theta_8]^T$, where $\theta_1, \theta_2, \ldots, \theta_8$ can be selected easily by intuitive reasoning. For example, if $x_i$ and $b_i$ are 0 and 1, respectively, and if the condition $f(\mathbf{x}) \ge f(\mathbf{b})$ is false:

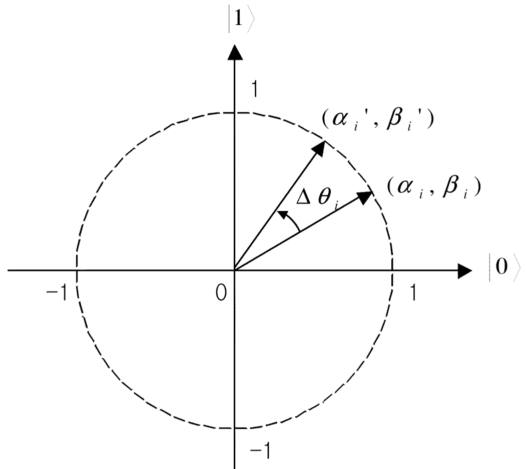


Fig. 2. Polar plot of the rotation gate for Q-bit individuals.

TABLE I
LOOKUP TABLE OF $\Delta\theta_i$, WHERE $f(\cdot)$ IS THE PROFIT, AND $b_i$ AND $x_i$ ARE THE $i$TH BITS OF THE BEST SOLUTION $\mathbf{b}$ AND THE BINARY SOLUTION $\mathbf{x}$, RESPECTIVELY. IN THE KNAPSACK PROBLEM, $\theta_1 = 0, \theta_2 = 0, \theta_3 = 0.01\pi$, $\theta_4 = 0, \theta_5 = -0.01\pi, \theta_6 = 0, \theta_7 = 0, \theta_8 = 0$ WERE USED

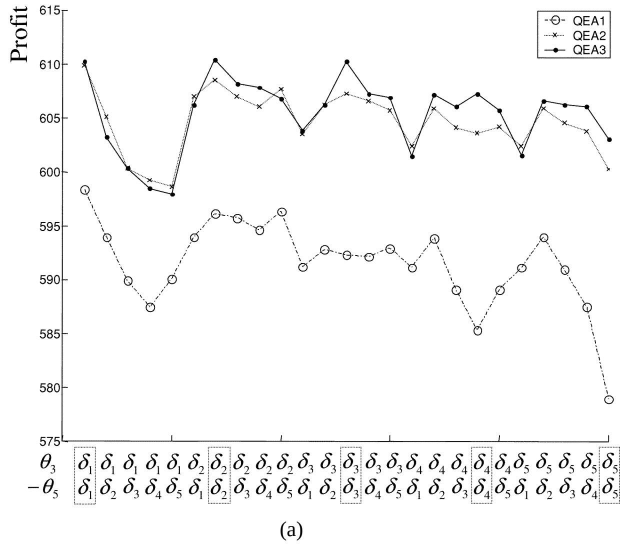
| $x_i$ | $b_i$ | $f(\mathbf{x}) \ge f(\mathbf{b})$ | $\Delta\theta_i$ |
|---|---|---|---|
| 0 | 0 | *false* | $\theta_1$ |
| 0 | 0 | *true* | $\theta_2$ |
| 0 | 1 | *false* | $\theta_3$ |
| 0 | 1 | *true* | $\theta_4$ |
| 1 | 0 | *false* | $\theta_5$ |
| 1 | 0 | *true* | $\theta_6$ |
| 1 | 1 | *false* | $\theta_7$ |
| 1 | 1 | *true* | $\theta_8$ |

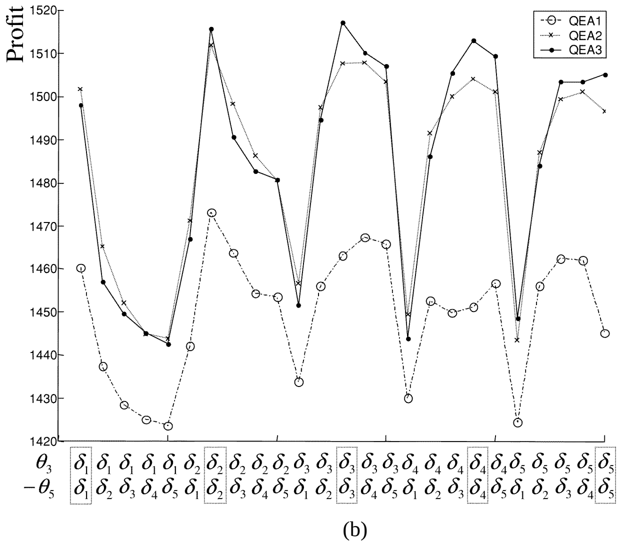
i) if the Q-bit is located in the first or the third quadrant in Fig. 2, $\theta_3$, the value of $\Delta\theta_i$ is set to a positive value to increase the probability of the state $|1\rangle$;
ii) if the Q-bit is located in the second or the fourth quadrant, $-\theta_3$ should be used to increase the probability of the state $|1\rangle$.

If $x_i$ and $b_i$ are 1 and 0, respectively, and if the condition $f(\mathbf{x}) \ge f(\mathbf{b})$ is false:

i) if the Q-bit is located in the first or the third quadrant, $\theta_5$ is set to a negative value to increase the probability of the state $|0\rangle$;
ii) if the Q-bit is located in the second or the fourth quadrant, $-\theta_5$ should be used to increase the probability of the state $|0\rangle$.

If it is ambiguous to select a positive or a negative number for the values of the angle parameters, it is recommended to set the values to 0. In the knapsack problem, $\theta_3 = 0.01\pi$, $\theta_5 = -0.01\pi$, and 0 for the rest were used. The magnitude of $\Delta\theta_i$ has an effect on the speed of convergence, but if it is too big, the solutions may diverge or converge prematurely to a local optimum. The values from $0.001\pi$ to $0.05\pi$ are recommended for the magnitude of $\Delta\theta_i$, although they depend on the





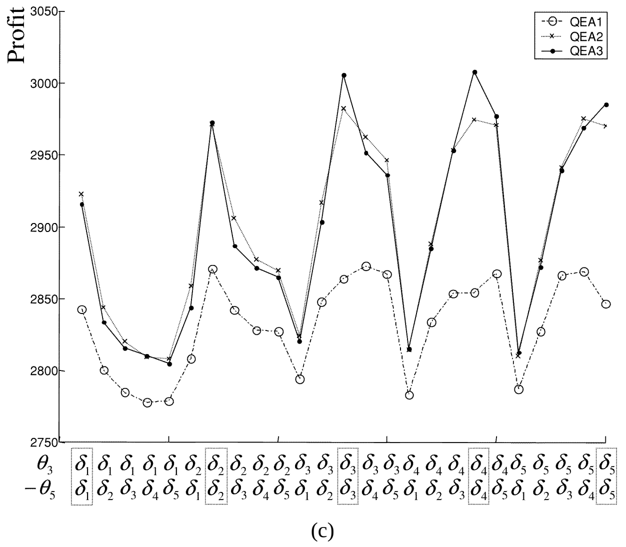


Fig. 3. Best profits of QEAs on the knapsack problems with (a) 100, (b) 250, and (c) 500 items to find good parameter settings of $\theta_3$ and $\theta_5$ of Table I. The vertical axis is the best profit averaged over 30 runs, and the horizontal axis is the parameter settings of ordered pairs of $\theta_3$ and $-\theta_5$. $\delta_1$, $\delta_2$, $\delta_3$, $\delta_4$, and $\delta_5$ are $0.0025\pi$, $0.005\pi$, $0.01\pi$, $0.02\pi$, and $0.05\pi$, respectively.

problems. The sign of $\Delta\theta_i$ determines the direction of convergence. The verification of the angle selection will be presented in Section IV-A.

## B. GA Methods for the Knapsack Problem

Three types of GA methods are described and tested for the knapsack problem: GAs based on penalty functions, GAs based on repair methods, and GAs based on decoders [20].

In these GAs based on penalty functions, a binary string of the length $m$ represents a chromosome $\mathbf{x}$ to the problem. The profit $f(\mathbf{x})$ of each string is determined as

$$f(\mathbf{x}) = \sum_{i=1}^{m} p_i x_i - Pen(\mathbf{x})$$

where $Pen(\mathbf{x})$ is a penalty function. There are several possible strategies for assigning the penalty function [21], [22]. Two types of penalties are considered, such as logarithmic penalty and linear penalty

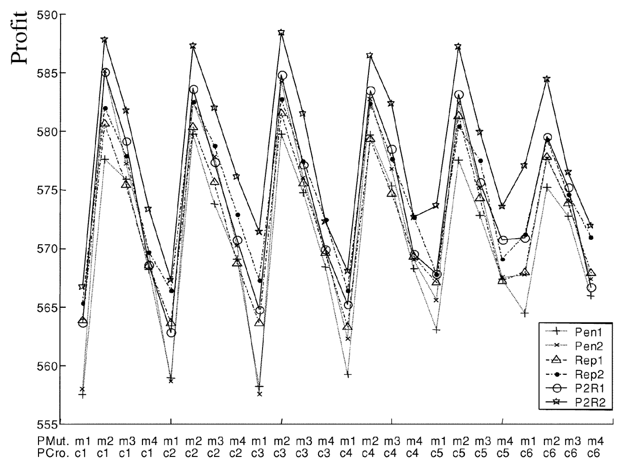
$$Pen_1(\mathbf{x}) = \log_2\left(1 + \rho\left(\sum_{i=1}^{m} w_i x_i - C\right)\right)$$
$$Pen_2(\mathbf{x}) = \rho\left(\sum_{i=1}^{m} w_i x_i - C\right)$$

where $\rho$ is $\max_{i=1\cdots m}\{p_i/w_i\}$.

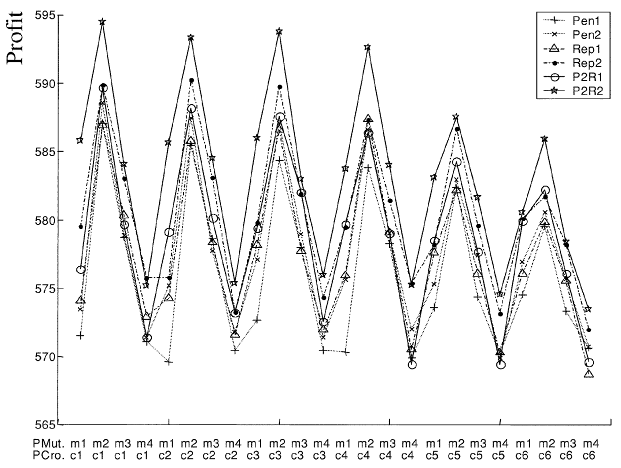
In GAs based on repair methods, the profit $f(\mathbf{x})$ of each string is determined as

$$f(\mathbf{x}) = \sum_{i=1}^{m} p_i x_i'$$

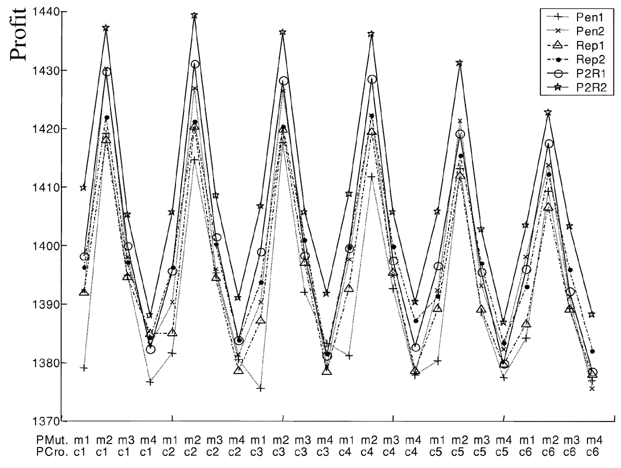
where $\mathbf{x}'$ is a repaired vector of the original vector $\mathbf{x}$. Original chromosomes are replaced with a 5% probability in the experiment. The two repair algorithms considered here differ only in selection procedure, which chooses an item for removal from the knapsack:



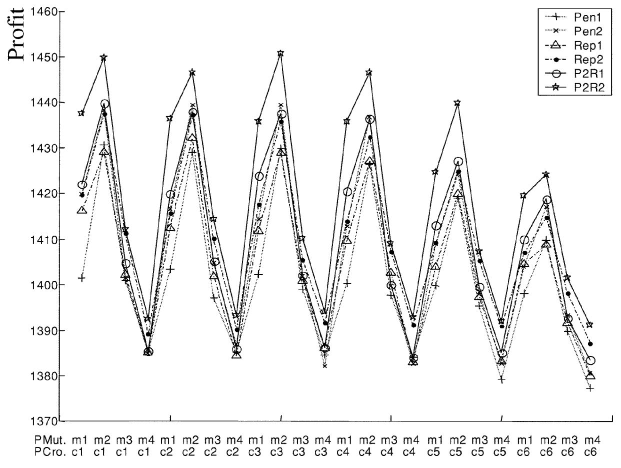
(a)



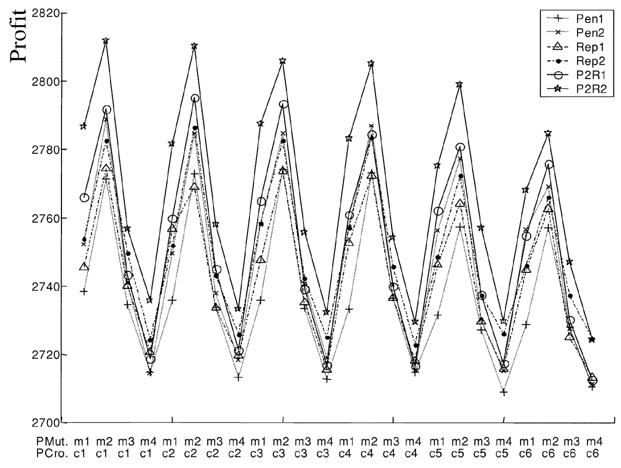
(b)



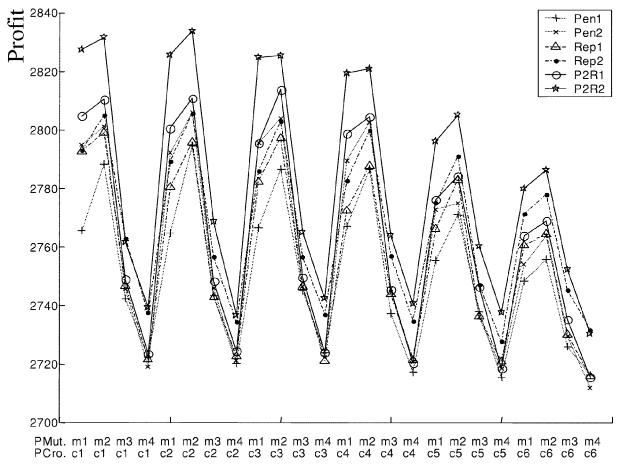
(c)



(d)



(e)



(f)

Fig. 4. Comparison of CGAs on the knapsack problems with 100, 250, and 500 items to find good parameter settings. The vertical axis is the best profit averaged over 30 runs, and the horizontal axis is the parameter settings of ordered pairs of the probabilities of mutation (PMut.) and crossover (PCro.). m1–m4 are 0.001, 0.01, 0.05, and 0.1, and c1–c6 are 0.001, 0.01, 0.05, 0.1, 0.3, and 0.5. (a) Population size 10 (100 items). (b) Population size 50 (100 items). (c) Population size 10 (250 items). (d) Population size 50 (250 items). (e) Population size 10 (500 items). (f) Population size 50 (500 items).

$Rep_1$ (random repair): The selection procedure selects a random element from the knapsack.

$Rep_2$ (greedy repair): All items in the knapsack are sorted in the decreasing order of their profit to weight ratios. The selection procedure always chooses the last item for deletion.

A possible decoder for the knapsack problem is based on an integer representation. Each chromosome is a vector of $m$ integers; the $i$th component of the vector is an integer in the range from 1 to $m-i+1$. The ordinal representation references a list $L$ of items; a vector is decoded by selecting appropriate item from the current list.

$Dec$ (random decoding): The build procedure creates a list $L$ of items such that the order of items on the list corresponds to the order of items in the input file which is random.

## C. Experimental Results

In all experiments, strongly correlated sets of data were considered

$$w_i = \text{uniformly random}\,[1,\,10]$$
$$p_i = w_i + 5$$

and the following average knapsack capacity was used:

$$C = \frac{1}{2}\sum_{i=1}^{m} w_i.$$

The data were unsorted. Three knapsack problems with 100, 250, and 500 items were considered.

The population sizes of QEA1, QEA2, and QEA3 were set to 1, 10, and 10, respectively. The global migration period in generation of QEA2 was 1 and that of QEA3 was 100. In QEA2, only global migration was used, and in QEA3, both global and local migrations were used. The local migration was implemented between each pair of neighboring solutions in $B(t)$ every generation. Fig. 3 shows the results of QEA1, QEA2, and QEA3 on the knapsack problems with 100, 250, and 500 items for finding good parameter settings of $\theta_3$ and $\theta_5$ of the lookup table. The values of $0.0025\pi$, $0.005\pi$, $0.01\pi$, $0.02\pi$, and $0.05\pi$ were tested for $\theta_3$ and $-\theta_5$. All the best profits were averaged over 30 runs, and the maximum number of generations was 1000. It should be noted that the results of the cases with the same value of $\theta_3$ and $-\theta_5$ were better than others. From the results, the values of $0.01\pi$ and $-0.01\pi$ were selected for $\theta_3$ and $\theta_5$, respectively.

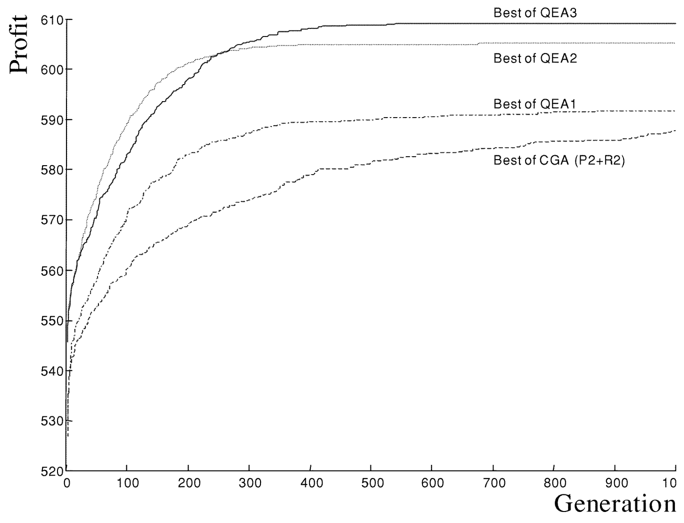
The population sizes of conventional GAs (CGAs) were 10 and 50. To discover good parameter settings of CGAs, the values of 0.001, 0.01, 0.05, and 0.1 for mutation and values of 0.001, 0.01, 0.05, 0.1, 0.3, and 0.5 for two-point crossover were tried on six CGAs: $Pen1$, $Pen2$, $Rep1$, $Rep2$, $Pen2 + Rep1$, and $Pen2 + Rep2$ ($Dec$ was not included in these experiments for finding parameters, since it took a long time to evolve and had worse performance as compared to other CGAs). $Pen2 + Rep1$ and $Pen2 + Rep2$ were designed by using a linear penalty function and random repair algorithm, and a linear penalty function and greedy repair algorithm, respectively. That is, 288 experiments per problem were tried (24 parameter settings $\times$ 6 CGAs $\times$ 2 population sizes). Fig. 4 shows the results of six CGAs on the knapsack problems with 100, 250, and 500 items to find good parameter settings. All the best profits were averaged over 30 runs, and the maximum number of generations was 1000. From Fig. 4, we could select (0.01, 0.001), (0.01, 0.01), and (0.01, 0.05) as ordered pairs of mutation and crossover probabilities that gave the maximum profit. The value of 0.01 for both mutation and crossover probabilities was selected for CGAs.

As a performance measure of the algorithms, we collected the best solution found within 1000 generations over 30 runs, and we checked the elapsed time per run, which is summarized in Table II, where only $Pen2 + Rep2$ ($P2R2$) among CGAs was shown because it outperformed all other CGAs. As Table II shows, QEAs yielded much better results compared to $P2R2$, except the results of $P2R2$ (50) and $QEA1$ with 100 items which is a relatively simple one compared to the other cases (250 and 500 items). The results show that QEAs perform well even with a small population. In the cases of 250 and 500 items, $QEA1$ found the better solutions within a short span of time compared to CGAs.

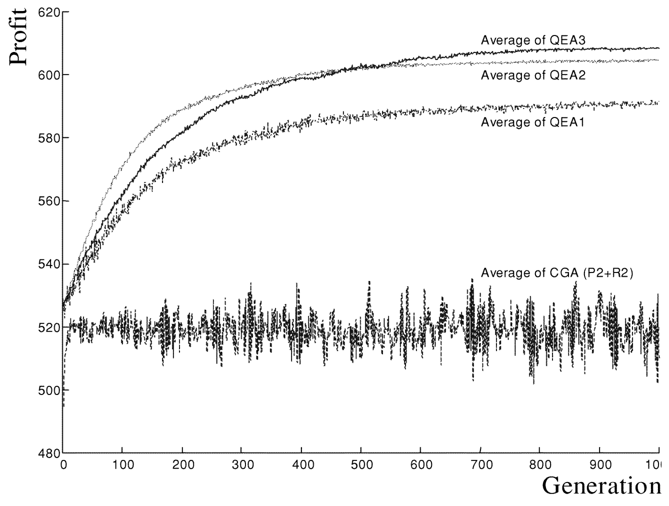
TABLE II
EXPERIMENTAL RESULTS OF THE KNAPSACK PROBLEM: THE NUMBER OF ITEMS 100, 250, AND 500, THE MAXIMUM NUMBER OF GENERATIONS 1000, THE NUMBER OF RUNS 30. THE PARENTHESIZED VALUES ARE THE POPULATION SIZES. $P2R2$ MEANS THE ALGORITHM IMPLEMENTED BY $Pen2 + Rep2$, AND $b.$, $m.$, AND $w.$ MEAN BEST, MEAN, $mean$, AND WORST, RESPECTIVELY. $t$ (SECONDS/RUN) AND $\sigma$ REPRESENT THE ELAPSED TIME PER RUN AND THE STANDARD DEVIATION, RESPECTIVELY

| | | CGAs | | QEAs | | |
|---|---|---|---|---|---|---|
| | | $P2R2$ (10) | $P2R2$ (50) | $QEA1$ (1) | $QEA2$ (10) | $QEA3$ (10) |
| 100 | b. | 597.6 | 602.2 | 597.7 | 612.7 | 612.7 |
| | m. | 587.8 | 593.6 | 591.8 | 606.3 | 609.5 |
| | w. | 577.6 | 582.6 | 582.5 | 597.7 | 607.6 |
| | $\sigma$ | 5.227 | 4.958 | 4.840 | 3.308 | 2.404 |
| | t | 0.154 | 0.786 | 0.021 | 0.199 | 0.203 |
| 250 | b. | 1455.0 | 1472.5 | 1480.2 | 1515.2 | 1525.2 |
| | m. | 1436.7 | 1452.4 | 1464.5 | 1508.1 | 1518.7 |
| | w. | 1415.2 | 1430.1 | 1445.1 | 1495.2 | 1515.2 |
| | $\sigma$ | 11.377 | 10.324 | 9.554 | 5.427 | 2.910 |
| | t | 0.357 | 1.804 | 0.055 | 0.531 | 0.558 |
| 500 | b. | 2828.1 | 2856.1 | 2899.7 | 3004.6 | 3025.8 |
| | m. | 2807.2 | 2831.0 | 2876.4 | 2980.8 | 3008.0 |
| | w. | 2781.0 | 2810.1 | 2836.2 | 2966.3 | 2996.1 |
| | $\sigma$ | 14.142 | 11.264 | 12.832 | 9.411 | 8.039 |
| | t | 0.706 | 3.559 | 0.117 | 1.212 | 1.258 |

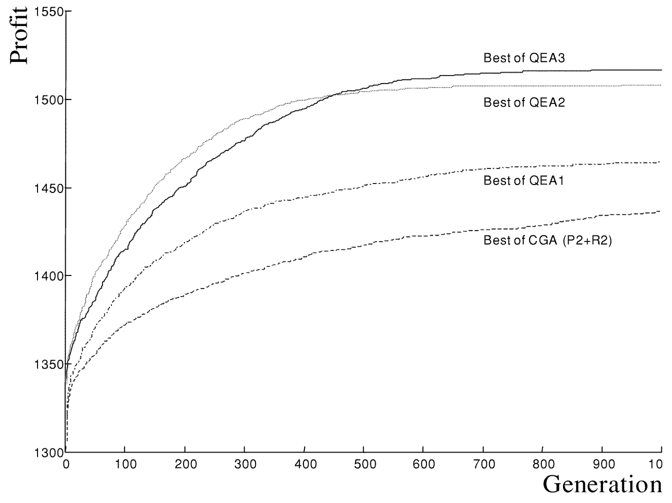
Fig. 5 shows the progress of the mean of best profits and the mean of average profits of the population found by QEA1, QEA2, QEA3, and CGA ($P2 + R2$) over 30 runs for 100, 250, and 500 items. QEAs perform significantly better than CGAs in terms of convergence rate and profit amount. QEA1 shows a slower convergence rate than QEA2 and QEA3 due to its single population size. QEAs show a faster convergence rate than CGA. QEA's final profits are much larger than CGA's in 1000 generations. The tendency of convergence rate can be shown clearly in the results of the mean of average profits of population. In the beginning, convergence rates of all the algorithms increased. However, CGA maintained a nearly constant profit due to its premature convergence. QEAs displayed no premature convergence in average profits until 1000 generations, which is a common feature of CGAs. In particular, in the results of QEA2 and QEA3, it should be mentioned that QEA3 initially shows a slower convergence rate than QEA2. However, QEA3 outperforms QEA2 in profits after about 500 generations, since QEA3 with a global migration process every 100 generations and a local migration process every generation can increase the population diversity.



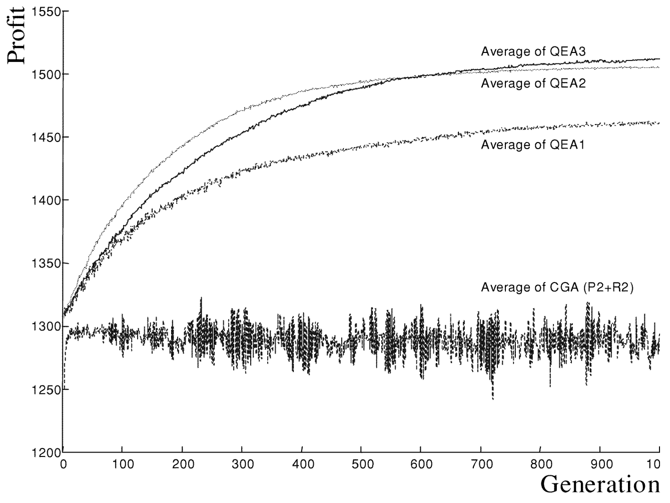
(a)



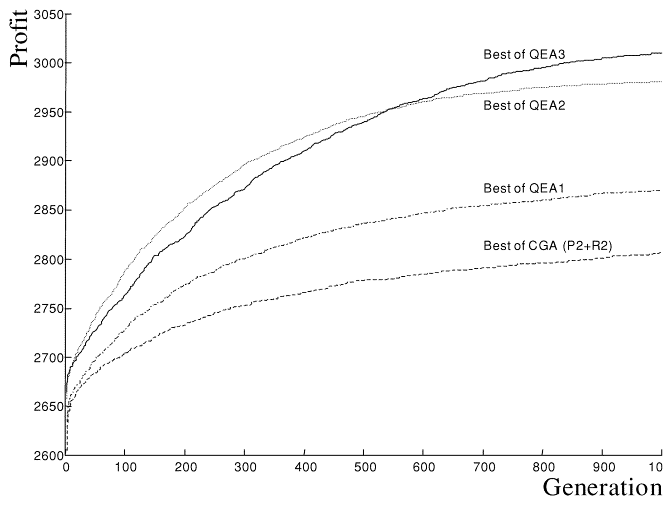
(b)



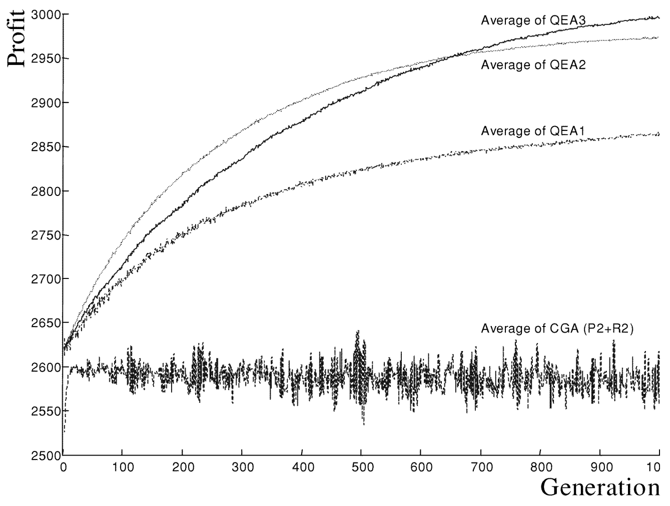
(c)



(d)

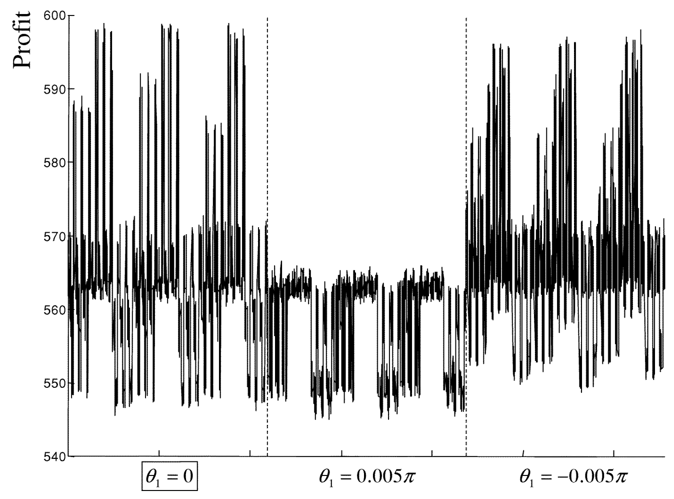


(e)

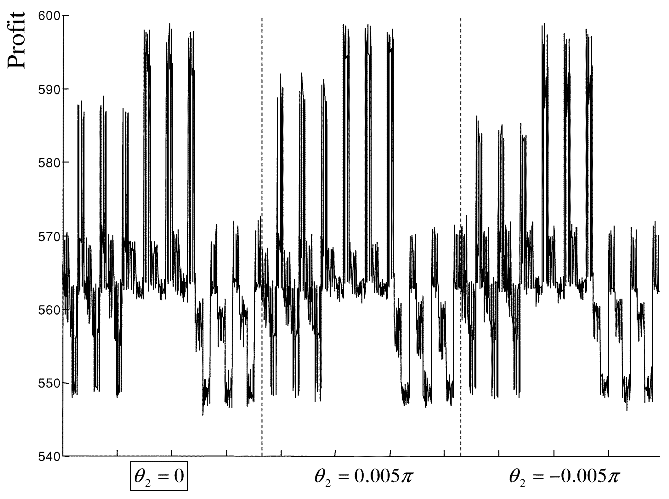


(f)

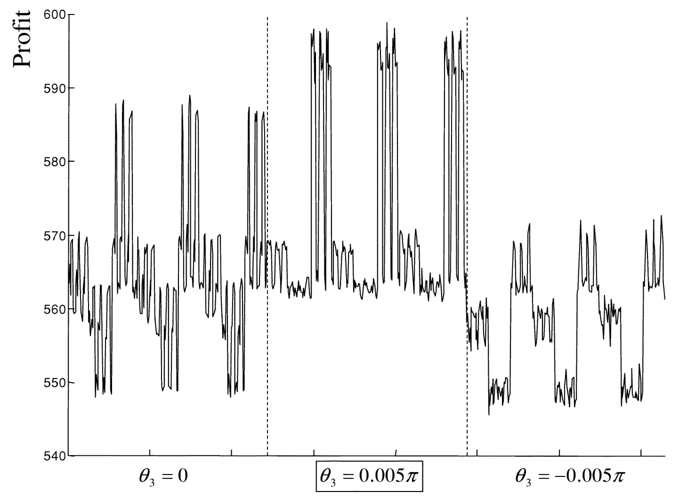
Fig. 5. Comparison of QEAs and CGA on the knapsack problem. The CGA is $Pen2 + Rep2$, and its population size is 10. The vertical axis is the profit value of knapsack, and the horizontal axis is the number of generations. The best profits and the average profits were averaged over 30 runs. (a) Best profits (100 items). (b) Average profits (100 items). (c) Best profits (250 items). (d) Average profits (250 items). (e) Best profits (500 items). (f) Average profits (500 items).



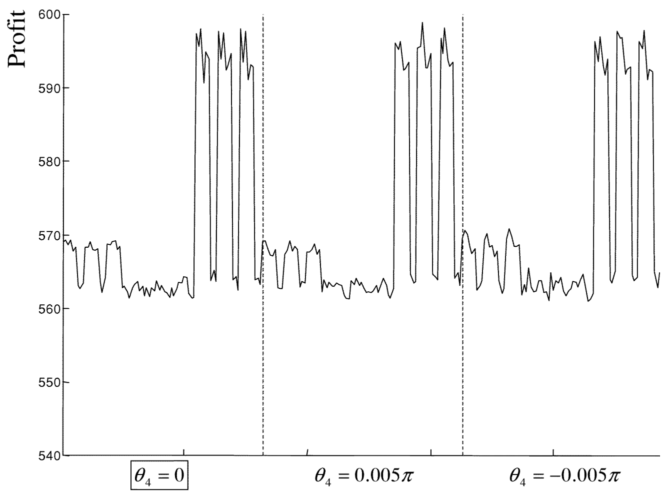
(a)



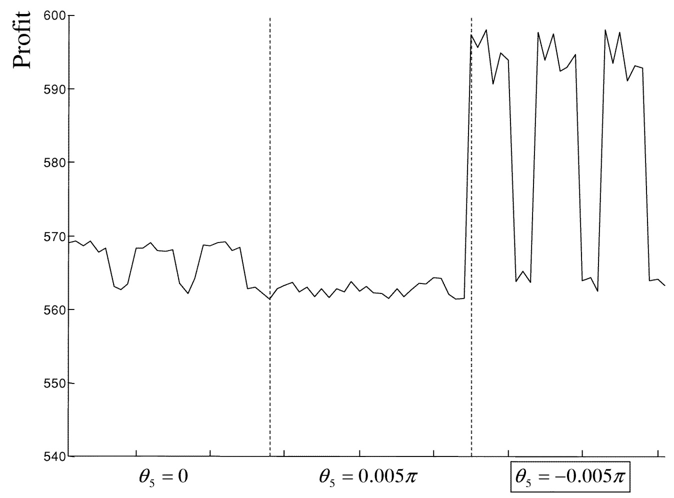
(b)



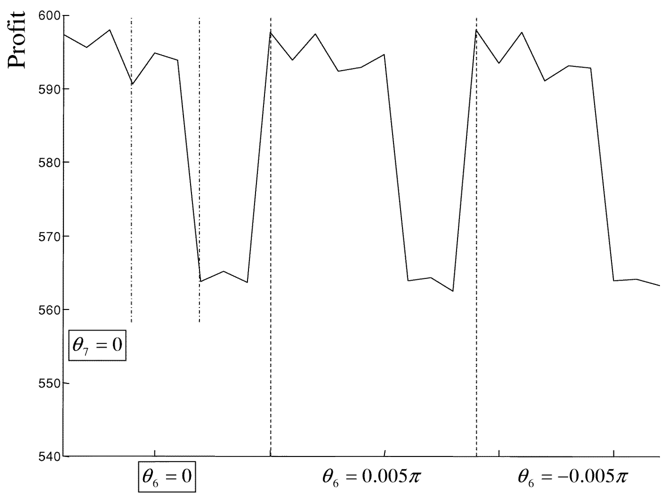
(c)



(d)



(e)



(f)

Fig. 6. Best profits of QEA1 on the knapsack problem with 100 items to find proper signs of the angle parameters of Table I. The vertical axis is the best profit averaged over 30 runs, and the horizontal axis is the parameter settings of the angle values. $*$ could be set to 0, $0.005\pi$, and $-0.005\pi$. $p$ and $n$ were set to $0.005\pi$ and $-0.005\pi$, respectively. (a) $\Theta = [* * * * * * * *]^T$. (b) $\Theta = [0 * * * * * * *]^T$. (c) $\Theta = [0\,0 * * * * * *]^T$. (d) $\Theta = [0\,0\,p * * * * *]^T$. (e) $\Theta = [0\,0\,p\,0 * * * *]^T$. (f) $\Theta = [0\,0\,p\,0\,n * * *]^T$.

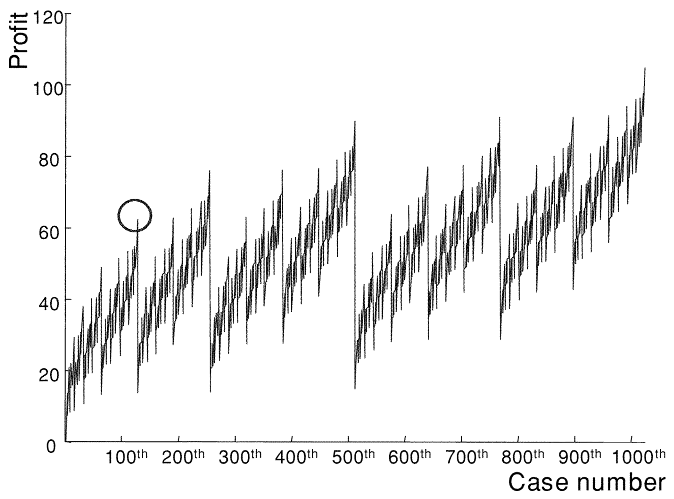


Fig. 7. Profit values of 1024 cases in the knapsack problem with ten items obtained by a simple calculation. The vertical axis is the profit values of the knapsack, and the horizontal axis is the number of 1024 cases selected as a subset from ten items. The best profit satisfying the capacity constraint is marked with O.

## IV. Characteristics of QEA

In this section, we verify the selection of the angle parameters for the rotation gate, and investigate the characteristics of the proposed QEA.

### A. Verification of the Angle Selection

In Section III, it was suggested to set a positive number for $\theta_3$, a negative number for $\theta_5$, and 0 for the rest of the angle parameters in $\Theta$ of Table I for the knapsack problem. To verify this intuitive reasoning, an experiment of QEA1 on the knapsack problem with 100 items was tried. The maximum number of generations was 1000. The values of 0, $0.005\pi$, and $-0.005\pi$ were used for each of the eight angle parameters. That is, $3^8$ cases of $\Theta$ were tried. Fig. 6(a)–(f) shows the experimental results carried out step by step to find proper signs $(0, +, -)$ of the angle parameters: (a) $3^8$ cases of $\Theta$, (b) $3^7$ cases of $\Theta$ when $\theta_1$ was selected as 0, (c) $3^6$ cases of $\Theta$ when both of $\theta_1$ and $\theta_2$ were selected as 0, (d) $3^5$ cases of $\Theta$ when both of $\theta_1$ and $\theta_2$ were 0, and $\theta_3$ was selected as a positive number, $0.005\pi$, (e) $3^4$ cases of $\Theta$ when both of $\theta_1$ and $\theta_2$ were 0, $\theta_3$ was $0.005\pi$, and $\theta_4$ was selected as 0, and (f) $3^3$ cases of $\Theta$ when both of $\theta_1$ and $\theta_2$ were 0, $\theta_3$ was $0.005\pi$, $\theta_4$ was 0, and $\theta_5$ was selected as a negative number, $-0.005\pi$. For the results of $\theta_2$, $\theta_4$, $\theta_6$, and $\theta_8$, that is, the cases in which $f(\mathbf{x}) \geq f(\mathbf{b})$ is true, it is worthwhile to mention that the values of $\theta_2$, $\theta_4$, $\theta_6$, and $\theta_8$ had little effect on the results as shown in Fig. 6(b), (d), and (f), respectively. It means that $\theta_2$, $\theta_4$, $\theta_6$, and $\theta_8$ can be set to any one among 0, $0.005\pi$, and $-0.005\pi$. In the results of the cases in which $f(\mathbf{x}) \geq f(\mathbf{b})$ is false, the values of 0, $0.005\pi$, $-0.005\pi$, and 0 for $\theta_1$, $\theta_3$, $\theta_5$, and $\theta_7$, respectively, made better solutions. From these experimental results, $\Theta$ could be assigned as $[0 * p * n * 0 *]^T$, where $*$ is one of (0, $p$, and $n$), $p$ is a positive number, and $n$ is a negative number. This is much the same as the intuitive reasoning mentioned in Section III.

### B. Investigation of the Characteristics of QEA

A simple knapsack problem with only ten items was considered to investigate the characteristics of QEA. Strongly correlated sets of data and the average knapsack capacity were used as in Section III-C. While selecting a subset from ten items, there exist $2^{10}$ cases. By a simple calculation we could obtain the profit values of 1024 cases in the knapsack problem shown in Fig. 7. In this problem, the best profit satisfying the capacity constraint was 62.192 938 at the 127th. The solutions with larger profits than the 127th one violated the capacity constraint. Now, to investigate the characteristics of QEA, a single Q-bit individual (QEA1) was used. A rotation gate and the parameter settings were the same as those of the experiments in Section III-C. Fig. 8 shows the probabilities of 1024 solutions using the Q-bit individual at generations, 10, 20, 30, 40, 50, 100, 200, and 300. Since all the possible solutions of the Q-bit individual are initialized with the same probability as described in (7), we have a probability of 0.001 ($1/\sqrt{2^{10}}^2 = 1/2^{10}$) for each solution which is shown in Fig. 8(a)–(c) as a horizontal line. It means that QEA initially starts with a random search.

With regard to the result at generation 10, it is worthwhile to mention that the probabilities of 1024 solutions had a pattern similar to the profit distribution of Fig. 7. It means that the only one Q-bit individual was able to represent 1024 cases similarly. At generation 20, solutions with larger probability appeared. At generation 30–50, the probabilities of the solutions with larger profit increased on a large scale. At generation 100, however, all the peak values decreased except the peaks of the better solutions. The same feature was obtained at generation 200. At generation 300, the probability of the best solution was over 0.9, and those of the other solutions were around 0. It means that the Q-bit individual had almost converged to the best solution.

The results above can be summarized in the following. Initially, QEA starts with a random search. At generation 10, the distribution of the probabilities of all the solutions becomes similar to the profit distribution in Fig. 7. As the probabilities of the solutions with larger profit increase, QEA starts a local search. Finally, the probability of the best Q-bit individual converges to 1. It means that QEA starts with a global search and changes automatically into a local search because of its inherent probabilistic mechanism, which leads to a good balance between exploration and exploitation. As the probabilistic representation is used, the termination condition can be given by the probability of the best solution such as $\mathrm{Prob}(\mathbf{b}) > \gamma$, where $0 < \gamma < 1$. $\mathrm{Prob}(\mathbf{b})$ is calculated by the multiplication of the probabilities of all the Q-bits in the Q-bit individual. For example, consider $\mathbf{b} = 1010$. Then, the probability of $\mathbf{b}$ can be obtained by $\mathrm{Prob}(\mathbf{b}) = |\beta_1|^2|\alpha_2|^2|\beta_3|^2|\alpha_4|^2$. Thus, the termination condition, **while** ($t < \mathrm{MAX\_GEN}$) can be replaced by **while** ($\mathrm{Prob}(\mathbf{b}) < \gamma$).

## V. Conclusions

This paper proposed a novel QEA, inspired by the concept of quantum computing. A Q-bit individual was defined as a string of Q-bits for the probabilistic representation. To introduce

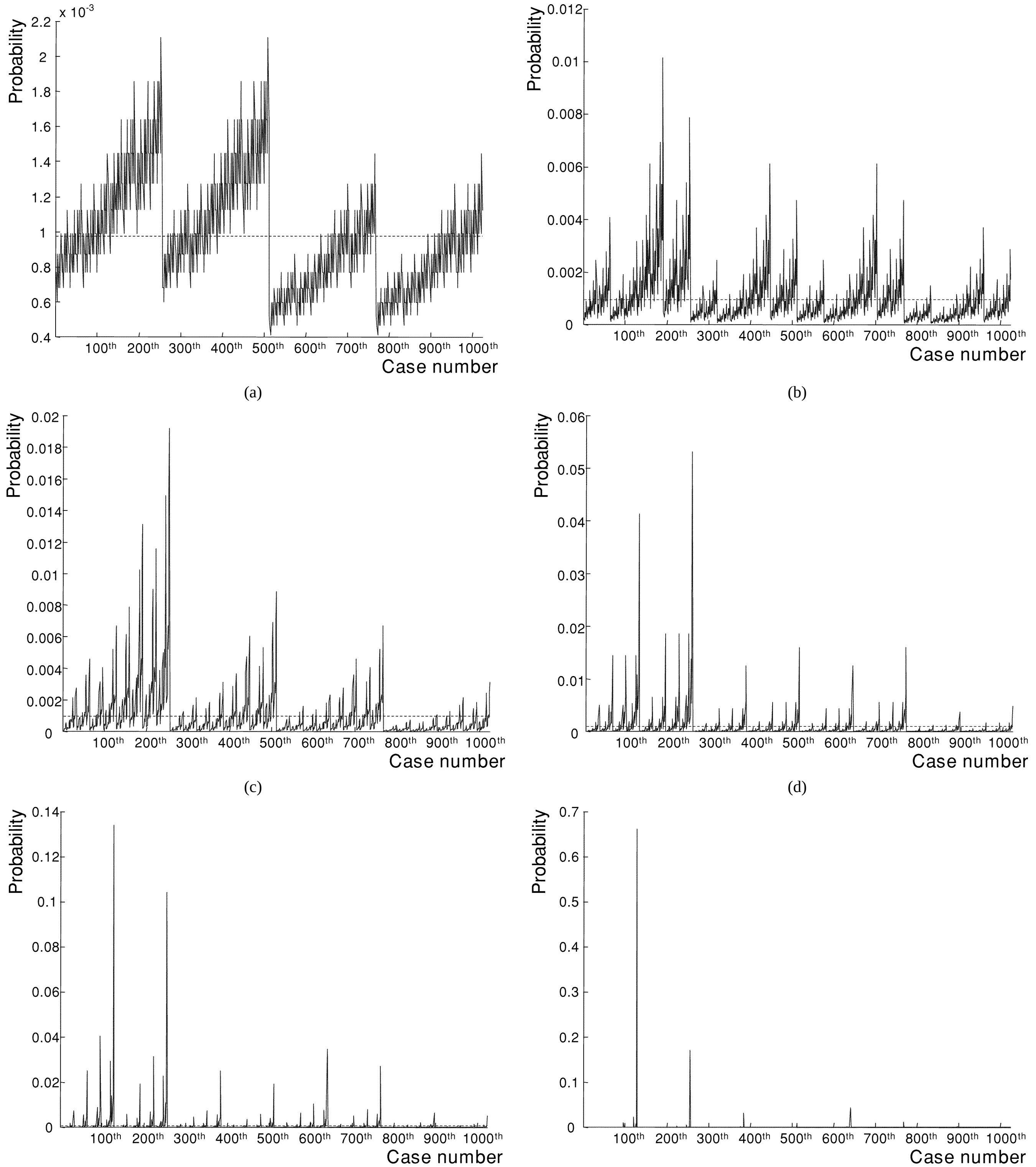
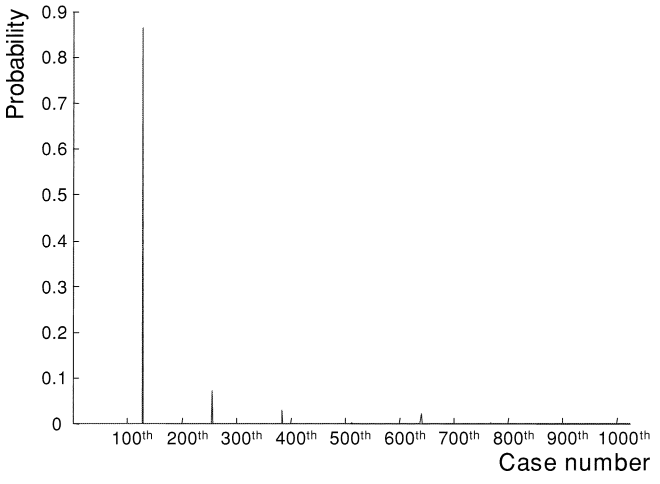


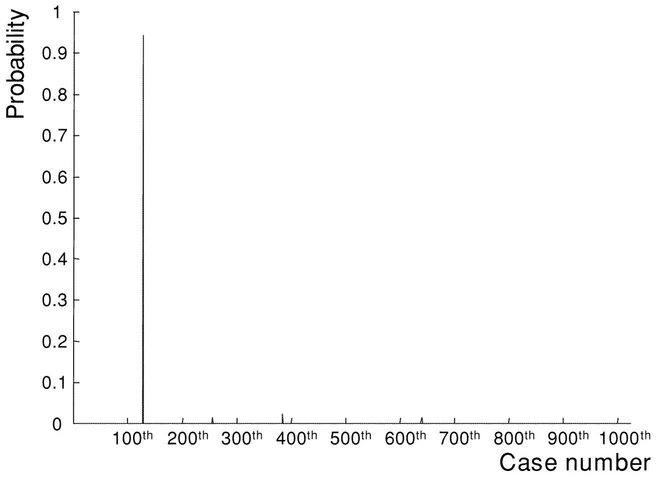
Fig. 8. Probabilities of all solutions using a Q-bit individual. The vertical axis is the probability of the solution, and the horizontal axis is the number of 1024 cases selected as a subset from ten items. (a) Generation 10. (b) Generation 20.(c) Generation 30. (d) Generation 40. (e) Generation 50. (f) Generation 100.

the variation to the Q-bit individual, a Q-gate is designed as a variation operator. The proposed QEA is characterized by the Q-bit representation for the population diversity, the observation process for producing a binary string from the Q-bit individual, the update process for driving the individuals toward better solutions by the Q-gate, the migration process for more variation of the probabilities of the Q-bit individuals, and the termination condition which can be given by the probability of the best solution.

The knapsack problem was considered as an application example to investigate the performance of QEA. In the experimental results, QEA performed quite well, even with only one



(g)



(h)

Fig. 8. *(Continued.)* Probabilities of all solutions using a Q-bit individual. The vertical axis is the probability of the solution, and the horizontal axis is the number of 1024 cases selected as a subset from ten items. (g) Generation 200. (h) Generation 300.

Q-bit individual. The characteristics of QEA could be verified by the simple knapsack problem with only ten items. The results demonstrated the effectiveness and the applicability of QEA to the combinatorial optimization problem.

## Acknowledgment

The authors would like to thank Dr. D. B. Fogel and the anonymous reviewers for their careful reading of this paper and for their helpful and constructive comments.

## References

[1] A. S. Fraser, "Simulation of genetic systems by automatic digital computers," *Aust. J. Biol. Sci.*, vol. 10, pp. 484–491, 1957.

[2] H. J. Bremermann, "Optimization through evolution and recombination," in *Self-Organizing Systems*, M. C. Yovits, G. T. Jacobi, and G. D. Goldstine, Eds. Washington, DC: Spartan, 1962, pp. 93–106.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: Univ. Michigan Press, 1975.

[4] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[5] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. Stuttgart, Germany: Frommann-Holzbog, 1973.

[6] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.

[7] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *J. Statist. Phys.*, vol. 22, pp. 563–591, 1980.

[8] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proc. Roy. Soc. London*, vol. A 400, pp. 97–117, 1985.

[9] P. W. Shor, "Quantum computing," *Doc. Mathematica*, vol. Extra Volume ICM, pp. 467–486, [Online]. Available: http://east.camel.math.ca/ EMIS/ journals/ DMJDMV/ xvol-icm/ 00/ Shor.MAN.html, 1998.

[10] ——, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Foundations of Computer Science*. Piscataway, NJ: IEEE Press, Nov. 1994, pp. 124–134.

[11] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th ACM Symp. Theory of Computing*, 1996, pp. 212–219.

[12] ——, "Quantum mechanical searching," in *Proc. 1999 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, July 1999, vol. 3, pp. 2255–2261.

[13] L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy, "Finding a better-than-classical quantum AND/OR algorithm using genetic programming," in *Proc. 1999 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, July 1999, vol. 3, pp. 2239–2246.

[14] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation*. Piscataway, NJ: IEEE Press, May 1996, pp. 61–66.

[15] A. Narayanan, "Quantum computing for beginners," in *Proc. 1999 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, July 1999, vol. 3, pp. 2231–2238.

[16] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proc. 2000 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, July 2000, vol. 2, pp. 1354–1360.

[17] K.-H. Han, K.-H. Park, C.-H. Lee, and J.-H. Kim, "Parallel quantum-inspired genetic algorithm for combinatorial optimization problem," in *Proc. 2001 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, May 2001, vol. 2, pp. 1422–1429.

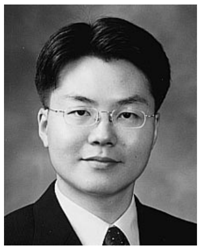
[18] T. Hey, "Quantum computing: An introduction," in *Computing & Control Engineering Journal*. Piscataway, NJ: IEEE Press, June 1999, vol. 10, no. 3, pp. 105–112.

[19] R. Hinterding, "Representation, constraint satisfaction and the knapsack problem," in *Proc. 1999 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, July 1999, vol. 2, pp. 1286–1292.

[20] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1999.

[21] J.-H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 129–140, July 1997.

[22] X. Yao, *Evolutionary Computation: Theory and Applications*. Singapore: World Scientific, 1999.



**Kuk-Hyun Han** received the B.S. and M.S. degrees in electrical engineering in 1997 and 1999, respectively, from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, where he is currently working toward the Ph.D. degree in electrical engineering and computer science.

His research interests are in the areas of quantum-inspired evolutionary computation, internet-based personal robot systems and intelligent control.

Mr. Han was the recipient of the Presidential Award at the 38th Nationwide Science Exhibition in 1992, the Certificate of Appreciation at the 1996 Micro-Robot World Cup Soccer Tournament (MiroSot'96), the runner-up Awards at S-MiroSot'97 and 2001 FIRA Cup China, and the Bronze Prize at the 7th Samsung HumanTech Thesis Competition in 2001.

**Jong-Hwan Kim** received his B.S., M.S., and Ph.D. degrees in electronics engineering from Seoul National University, Korea, in 1981, 1983 and 1987, respectively.

Since 1988, he has been with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, where he is currently a Professor. He was a Visiting Scholar at Purdue University and the University of California at Davis during 1992 and 2000, respectively. His current research interests are in the area of evolutionary multiagent robotic systems.

Dr. Kim currently serves as an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the *International Journal of Intelligent and Fuzzy Systems*. He was Guest Editor for the *Journal of Robotics and Autonomous Systems* (1996) and the *Journal of Intelligent Automation and Soft Computing* (2000). He was one of the co-founders of the Asia-Pacific Conference on Simulated Evolution and Learning, the General Chair of the 2001 Congress on Evolutionary Computation held in Seoul, Korea, and is currently serving FIRA (The Federation of International Robot-soccer Association) and IROC (The International Robot Olympiad Committee) as President. He was the recipient of the Choongang Young Investigator Award in 1988 from Choongang Memorial Foundation, the LG YonAm Foundation Research Fellowship in 1992, the Korean Presidential Award in 1997, and the SeoAm Foundation Research Fellowship in 1999. His name is included in the *Barons 500 Leaders for the New Century* as Founder of FIRA and the Robot Olympiad.