박 사 학 위 논 문

Doctoral Thesis

양자 개념을 도입한 진화 알고리즘

Quantum-inspired Evolutionary Algorithm

한 국 현 (韓 國 鉉 Han, Kuk Hyun)

전자전산학과 전기및전자공학전공

Department of Electrical Engineering and Computer Science

Division of Electrical Engineering

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

2003

양자 개념을 도입한 진화 알고리즘

Quantum-inspired Evolutionary Algorithm

Quantum-inspired Evolutionary Algorithm

Advisor : Professor Jong-Hwan Kim

by

Kuk-Hyun Han

Department of Electrical Engineering and Computer Science
Division of Electrical Engineering
Korea Advanced Institute of Science and Technology

A thesis submitted to the faculty of the Korea Advanced Institute of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical Engineering and Computer Science, Division of Electrical Engineering

Daejeon, Korea
2003. 6. 5.
Approved by

Professor Jong-Hwan Kim
Major Advisor

양자 개념을 도입한 진화 알고리즘

한 국 현

위 논문은 한국과학기술원 박사학위논문으로 학위논문심사 위원회에서 심사 통과하였음.

2003년 5월 21일

심사위원장 김 종 환 (인)

심사위원 박 규 호 (인)

심사위원 임 종 태 (인)

심사위원 탁 민 제 (인)

심사위원 홍 성 철 (인)

DEE
995393

한 국 현. Kuk-Hyun Han. Quantum-inspired Evolutionary Algorithm. 양자 개념을 도입한 진화 알고리즘. Department of Electrical Engineering and Computer Science. 2003. 127p. Advisor Prof. Jong-Hwan Kim. Text in English.

Abstract

This thesis 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 that drives the individuals toward better solutions. The termination condition of QEA is designed by defining a new measure on the convergence of Q-bit individuals. To analyze the characteristics of QEA, the theoretical analysis of the QEA algorithm as well as the effects of changing parameters of QEA are examined. In particular, some issues of QEA such as the analysis of changing the initial values of Q-bits, a novel variation operator H_ϵ gate, and a two-phase QEA (TPQEA) scheme are addressed to improve the performance of QEA. To demonstrate the effectiveness and applicability of QEA, 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 number of population, without premature convergence as compared to the conventional genetic algorithms. Moreover, through the experiments on numerical optimization problems, the superior performance of QEA is also verified. These results show that QEA can be applied to a class of numerical as well as combinatorial optimization problems.

Contents

1. Introduction

1.1 Background and motivation

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 methods and enumerative strategies, EAs are robust, global in operation, and may be applied generally without recourse to domain-specific heuristics, although their performance may be 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.

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, these components should be designed properly. In particular, in this thesis the representation and population dynamics are investigated to represent the individuals effectively to explore the search space with a smaller number of individuals (even with only one individual for real-time application) and to exploit the search space for a global solution within a short span of time, respectively. For these purposes, some concepts of quantum computing are adopted in the proposed evolutionary algorithm.

Quantum computing is a research area which includes concepts like quantum mechanical computers and quantum algorithms. Quantum mechanical computers were proposed in the early 1980s [7, 8] and their description was formalized in the late 1980s [9, 10]. Many efforts on quantum computers have progressed actively since the early 1990s because these computers were shown to be more powerful than digital computers for solving various specialized problems. There are well-known quantum algorithms such as Deutsch-Jozsa algorithm [11], Simon's algorithm [12], Shor's quantum factoring algorithm [13, 14], and Grover's database search algorithm [15, 16, 17]. Shor's algorithm finds the prime factors of an n-digit number in polynomial-time, while the best-known classical factoring algorithms require time $O(2^{n^{\frac{1}{3}}\log(n)^{\frac{2}{3}}})$ [18]. Grover's algorithm can find an item in an unsorted list of n items in $O(\sqrt{n})$ steps, while any classical algorithm needs to access the list a minimum of $0.5n$ times. If, for example, the speed of quantum or digital computer is 1 MIPS, Grover's algorithm can find the secret key of 56-bit string within 4 minutes in quantum computer without any factoring algorithms, while the classical algorithm can find it within $1,000$ years [19]. In particular, since the difficulty of the factoring problem is crucial for the security of the RSA cryptosystem [20] which is in widespread use today, interest in quantum computing is increasing [21].

Research on merging evolutionary computation and quantum computing has started since the late 1990s. It can be classified into two groups. One group

concentrates on generating new quantum algorithms using automatic programming techniques such as genetic programming [18, 22, 23]. The other concentrates on quantum-inspired evolutionary computing for a digital computer, and is a branch of study on evolutionary computation that is characterized by certain principles of quantum mechanics such as uncertainty, superposition, interference, etc. [24, 25, 26, 27].

Unlike other research areas, there has been relatively little work done in applying quantum computing to evolutionary algorithms. Quantum-inspired computing was introduced in [28]. In [24], a modified crossover operator which includes the concept of interference was introduced. In [25], a probabilistic representation and a novel population dynamics inspired by quantum computing were proposed. In [26], the applicability of quantum-inspired evolutionary algorithm to a parallel scheme, particularly, PC clustering, was verified successfully. In [27], the basic structure of quantum-inspired evolutionary algorithm (QEA) and its characteristics were formulated and analyzed, respectively. According to [27], the results (tested on the knapsack problem) of QEA were proved to be better than those of CGA (conventional genetic algorithm). In [29], a QEA-based disk allocation method (QDM) was proposed. According to the results, the average query response times of QDM were equal to or less than those of DAGA (disk allocation methods using GA), and the convergence speed of QDM was 3.2-11.3 times faster than that of DAGA. In [30], QEA was applied to a decision boundary optimization for face verification. The proposed face verification system was tested by face and non-face images extracted from AR face database [31]. Compared to the conventional PCA (principal components analysis) method improved results were achieved both in terms of the face verification rate and false alarm rate. Other research on quantum-inspired computing has also been investigated [32, 33].

With no connection to quantum computing, a number of evolutionary algorithms

that guide the exploration of the search space by building probabilistic models of promising solutions found have been introduced since the late 1990s [34]. These algorithms have shown to perform well on a variety of problems. In the population-based incremental learning (PBIL) which is a method of combining the mechanisms of a generational genetic algorithm with simple competitive learning [35], the solutions are represented by binary strings and the population of solutions is replaced with a probability vector. The compact genetic algorithm (cGA) [36] replaces the population with a single probability vector as in PBIL, however its modification method of the probability vector is different from PBIL. In the extended compact genetic algorithm (ECGA) [37], the variables are divided into a number of intact clusters which are manipulated as independent variables. The Bayesian optimization algorithm (BOA) [38] uses a more general class of distributions than ECGA. It incorporates methods for learning Bayesian networks and uses these to model the promising solutions and generate the new ones.

1.2 Research objectives and outlines

This research aims at proposing 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 any other 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 to be the smallest unit of information. A Q-bit individual is represented 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 is a better characterization of population diversity than any other representations. A Q-gate is also defined as a variation operator of

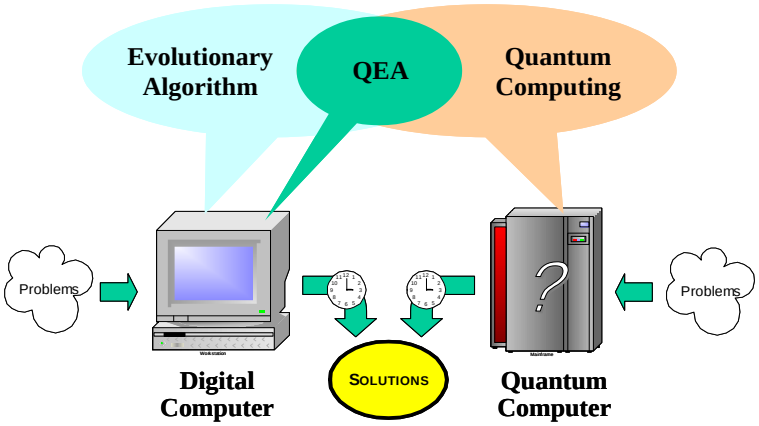


Figure 1.1: Quantum-inspired evolutionary algorithm (QEA).

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 the 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 digital computer as shown in Figure 1.1 [27, 39]. To demonstrate its performance, several experiments on a class of numerical and combinatorial optimization problems have been carried out. The results show that QEA performs better, even with a small population, without premature convergence as compared to the conventional evolutionary algorithms.

In Chapter 2, evolutionary computation is introduced briefly as a general outline. Then the history and basics of quantum computation are described. Qubit, quantum gate, superposition, and entanglement are regarded as the basics.

In Chapter 3, Q-bit and Q-bit individual are defined for the representation of

QEA. Then the basic structure of QEA is proposed and each step of QEA is described from a theoretical viewpoint. The knapsack problem is considered to demonstrate the applicability of QEA to a class of combinatorial optimization problems. The concrete procedure QEA for the knapsack problem is described and the experiments are carried out to demonstrate its performance in comparison to the conventional genetic algorithms. The empirical and theoretical analyses of QEA follow to investigate the characteristics of QEA. In particular, the termination criteria and the effects of changing parameters are also investigated.

In Chapter 4, the basic structure of QEA is extended for the improvement in its performance. The effects of changing the initial values of Q-bits are investigated, since the initial values can influence the performance of QEA. A novel variation operator H_ϵ gate is proposed to provide an attempt to escape effectively from many local optima. As an extended version of QEA, a two-phase QEA (TPQEA) scheme is also proposed by analyzing the effect of changing the initial values of Q-bits. In the first phase some promising initial values of Q-bits are searched, which will be used in the second phase. By employing the second phase, the performance of QEA can be increased for a class of optimization problems. To verify these issues, some experiments have been carried out.

In Chapter 5, several numerical and combinatorial optimization problems are picked up to demonstrate the effectiveness and applicability of QEA including the issues of the H_ϵ gate and the TPQEA scheme.

Finally in Chapter 6, conclusions on this thesis are presented. Further research scope is also discussed in detail.

2. Evolutionary Computation and Quantum Computation

2.1 Evolutionary computation

More than 45 years ago, a number of innovative researchers at different places in the US and Europe independently came up with the idea of mimicking mechanisms of biological evolution in order to develop powerful algorithms for problems of adaptation and optimization. Overviews of current state of the art in the field of evolutionary computation are given by Fogel [40] and Bäck [41].

EAs are based on computational models of fundamental evolutionary processes such as selection, recombination and mutation. Individuals, or current approximations, are encoded as strings composed over some alphabet(s), e.g. binary, integer, real-valued, etc., and an initial population is produced by randomly sampling these strings. Once a population is produced, it may be evaluated using an objective function which characterizes an individual's performance in the problem domain. The objective function is also used as the basis for selection, and determines how well an individual performs in its environment. A fitness value is then derived from the raw performance measure given by the objective function, and is used to bias the selection process. Highly fit individuals will be assigned a higher probability of being selected for reproduction than the individuals with a lower fitness value. Therefore, the average performance of individuals can be expected to increase as the best fit individuals are more likely to be selected for reproduction, and the less fit individuals are discarded. Note that individuals may be selected more than once in any

```
Procedure EA
begin
        t ← 0
        initialize P(t)
        evaluate P(t)
        while (not termination-condition) do
        begin
            t ← t + 1
            select P(t) from P(t − 1)
            reproduce pairs in P(t)
            mutate P(t)
            evaluate P(t)
        end
end
```

Figure 2.1: General evolutionary algorithm.

generation (iteration) of the EA.

Selected individuals are then reproduced, usually in pairs, through the application of genetic operators. These operators are applied to pairs of individuals with a given probability and result in new offsprings that contain materials exchanged from their parents. The offsprings are then further perturbed by mutation. These new individuals then make up the next generation. The processes of selection, reproduction and evaluation are then repeated until some termination criteria are satisfied, e.g. a certain number of generations completed, a mean deviation in the performance of individuals in the population is below a certain value or when a particular point in the search space is reached. The pseudo-code of a general evolutionary algorithm is shown in Figure 2.1.

T. Bäck described notations and definition of EAs in his book [41] as follows:

An *Evolutionary Algorithm*(EA) is defined as an 8-tuple

$$EA = (I, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda) \tag{2.1}$$

where $I = A_x \times A_s$ is the space of *individuals*, and A_x, A_s denote arbitrary sets. $\Phi : I \to \mathbb{R}$ denotes a *fitness function* assigning real values to individuals.

$$\Omega = \{\omega_{\Theta_1}, \cdots, \omega_{\Theta_z} | \omega_{\Theta_i} : I^\lambda \to I^\lambda\} \cup \{\omega_{\Theta_0} : I^\mu \to I^\lambda\} \tag{2.2}$$

is a set of probabilistic *genetic operator* ω_{Θ_i}, each of which is controlled by specific parameters summarized in the sets $\Theta_i \subset \mathbb{R}$.

$$s_{\Theta_s} : (I^\lambda \cup I^{\lambda+\mu}) \to I^\mu \tag{2.3}$$

denotes the *selection operator*, which may change the number of individuals from λ or $\mu + \lambda$ to μ, where μ, $\lambda \in \mathbb{N}$ and $\mu = \lambda$ is permitted. An additional set Θ_s or parameters may be used by the selection operator. μ is the number of parent individuals, while λ denotes the number of offspring individuals. Finally, $\iota : I^\mu \to \{True, Flase\}$ is a *termination criterion* for the EA, and the generation transition function $\Psi : I^\mu \to I^\mu$ describes the complete process of transforming a population into a subsequent one by applying genetic operators and selection.

The space of individuals may be arbitrarily complex, i.e. there are no restrictions on the structure of the sets A_x and A_s. Even the fitness function Φ may include some intermediate calculation steps, one of those always being evaluation of the objective function value which provides the basis of the fitness value. Whenever $\mu \neq \lambda$, the operator set Ω includes a distinguished operator $\omega_0 : I^\mu \to I^\lambda$ which serves to change population size forming λ offspring individuals from μ parents. While genetic operators are always probabilistic, selection may be probabilistic or completely deterministic. Both selection and genetic operators may be controlled by

Algorithm (Outline of an EA)
begin
$t \leftarrow 0$
$P(t) \leftarrow initialize(\mu)$
$\Phi(t) \leftarrow evaluate(P(t), \mu)$
while ($\iota(P(t), \Theta_\iota) \neq$ true) **do**
begin
$P'(t) \leftarrow recombine(P(t), \Theta_r)$
$P''(t) \leftarrow mutate(P'(t), \Theta_m)$
$\Phi(t) \leftarrow evaluate(P''(t), \lambda)$
$P(t+1) \leftarrow select(P''(t), \Phi(t), \mu, \Theta_s)$
$t \leftarrow t + 1$
end
end

Figure 2.2: Outline of an EA by T. Bäck.

some exogenous parameters. The termination criterion ι may range from arbitrarily complicated criteria - e.g., genotypic or phenotypic diversity of the population, relatively improvement of the best objective function value over subsequent generations - to rather simple ones, e.g., testing whether a specified number of generations is completed.

The description given above can be translated into a general algorithmic outline of an EA. Θ_m, Θ_r, Θ_ι, and Θ_s denote the parameters of mutation, recombination, termination, and genetic operators, respectively, and t denotes the generation counter. $P(t)$ and $\Phi(t)$ are the population and the fitness at generation t, respectively, and μ and λ denote the parent population size and offspring population size, respectively [42].

However, it should be noted that the notations of QEA proposed in this thesis are different from those given above.

The three main-stream methods of evolutionary computation are genetic algo-

	ES	EP	GAs
Representation	Real-valued	Real-valued	Binary-valued
Fitness is	Objective function value	Scaled objective function value	Scaled objective function value
Self-adaptation	Standard deviations and rotation angles	None (standard EP) variances (meta-EP)	None
Mutation	Gaussian, main operator	Gaussian, only operator	Bit-inversion, background operator
Recombination	Discrete and intermediate, sexual and panmictic	None	Crossover, only sexual, main operator
Selection	Deterministic	Probabilistic, extinctive	Probabilistic based on preservation

Table 2.1: Main characteristics of EAs.

rithms (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].

Genetic algorithms emphasize recombination (crossover) as the most important search operator and apply mutation with very small probability solely as a background operator. They also use a probabilistic selection operator (proportional selection) and often rely on a binary representation of individuals [43, 44, 45].

Evolution strategies use normally-distributed mutations to modify real-valued vectors and emphasize mutation and recombination as essential operators for searching in the search space and in the strategy parameter space at the same time. The selection operator is deterministic, and parent and offspring population sizes usually differ from each other [46].

Evolutionary programming emphasizes mutation and does not incorporate the

recombination of individuals. Similarly to evolution strategies, when approaching real-valued optimization problems, evolutionary programming also works with normally distributed mutations and extends the evolutionary process to the strategy parameters. The selection operator is probabilistic. Presently, most applications are reported for search spaces involving real-valued vectors, although the algorithm was originally developed to evolve finite-state machines [47].

The most important characteristics of EAs are summarized in Table 2.1 [48].

2.2 Quantum computation

Quantum computation is a research area that is based on the characteristics of quantum mechanics such as uncertainty, superposition, interference, and entanglement to process information through novel methods basically different from conventional techniques. Quantum computation is referred to as quantum information science, including quantum cryptography [49], quantum teleportation [50] as well as quantum computing. Quantum computing deals with two main topics: quantum computer and quantum algorithm.

2.2.1 History of quantum computation

The universal Turing machine (TM) is perhaps the most general computer possible, and all general purpose computers are approximations to it [51]. The universal TM can simulate any TM with perfect precision, where a TM in turn is a theoretical model that can simulate the execution of a single algorithm on a digital computer. However, R. Landauer pointed out that erasure of information is necessarily a dissipative process. His insight is that erasure always involves the compression of phase space, and so is irreversible [52]. In 1973, C. H. Bennett found that classical computation can be broken into a series of steps, each logically reversible, and this in turn allows physical reversibility of the computation [53]. P. Benioff proposed quan-

tum mechanical hamiltonian models of TMs which do not dissipate any energy and operate at the quantum limit in that the system (energy uncertainty)/(computation speed) is close to the limit given by the time-energy uncertainty principle [7, 54]. The models, however, are different from the concept of a current quantum computer. R. Feynman showed how a quantum system could be used to perform computations and could act as a simulator for probabilistically weighted quantum processes [8].

D. Deutsch showed that every finitely realizable physical system can be perfectly simulated by a universal quantum computer operating by finite means [9]. He also analyzed the role of quantum parallelism, and commented on the role of quantum complexity theory. A theory of quantum computational networks which is a generalization of the theory of quantum logic gates was also described by him [10]. In a paper with R. Jozsa, he described an algorithm that illustrates the power of quantum computation [11].

The real interest in exploring the bridge between physics and computation arose when quantum algorithms which improved over their classical counterparts were proposed [13, 12, 15]. In [13], P. Shor proposed a factoring algorithm for a quantum computer that finds the prime factors of a composite integer more efficiently than is possible with the known algorithms for a classical computer. Since the difficulty of the factoring problem is crucial for the security of a public key encryption system, interest in quantum computing is increasing. In [12], D. Simon presented an expected polynomial-time algorithm for a quantum computer that distinguishes between two reasonably natural classes of polynomial-time computable function. In [15], L. K. Grover proposed an algorithm which achieves quadratic speedup for the classic problem of database search. In the problem of a phone directory containing N names arranged in completely random order, for example, any classical algorithm (whether deterministic or probabilistic) needs to access a database a minimum of $0.5N$ times to find someone's phone number with a probability of 50%.

	Digital computer	Quantum computer
Information	Boolean (bit)	Boolean (qubit)
Information implementation	Voltage of wire	State of spin-$\frac{1}{2}$ system
Bit	Added or removed	Conserved
Reversibility	Irreversible	Conserved
Gate	Spatially arranged combination of transistors	Time-ordered unitary operators

Table 2.2: Differences between a digital computer and a quantum computer.

Using the Grover's algorithm, however, the desired phone number can be obtained in only $O(\sqrt{N})$ accesses to the database, since quantum mechanical systems can be in a superposition of states and simultaneously examine multiple names.

2.2.2 Basics of quantum computation

The smallest unit of digital information is a bit, which takes one of the two possible values $\{0, 1\}$. The corresponding unit of quantum information stored in a two-state quantum computer is called a quantum bit or qubit [55, 56]. A qubit may be in the '1' state, in the '0' state, or in any superposition of the two. It describes a state in the simplest possible quantum system. Table 2.2 shows the differences between a digital computer and a quantum computer [57].

The state of a qubit can be represented as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.4}$$

where α and β are complex numbers that specify the probability amplitudes of the corresponding states [58]. A qubit is a state in a two-dimensional Hilbert space that can take any value of the form (2.4). $|\alpha|^2$ 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

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.5}$$

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 [59].

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 NOT gate, Controlled NOT gate, Hadamard gate, Square root of NOT gate, Phase gate, Controlled Phase Shift gate, etc. [60]. A NOT gate is

$$U_{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

and its operation is shown as follows:

$$\begin{aligned} |0\rangle &\longrightarrow |1\rangle \\ |1\rangle &\longrightarrow |0\rangle. \end{aligned}$$

In Controlled NOT (CNOT) gate, the NOT operation is only operative when the state of the controlled qubit is ‘1’ state. The CNOT gate is

$$U_{CNOT}(1,2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and its operation is shown as follows:

$$|00\rangle \longrightarrow |00\rangle, \quad |01\rangle \longrightarrow |01\rangle,$$
$$|10\rangle \longrightarrow |11\rangle, \quad |11\rangle \longrightarrow |10\rangle.$$

A Hadamard gate is

$$U_H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

and its operation is shown as follows:

$$\begin{aligned} |0\rangle &\longrightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ |1\rangle &\longrightarrow \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned}$$

U_H transforms a basis vector into superpositions. A Square Root of NOT gate is as follows:

$$\begin{aligned} U_{\sqrt{NOT}} &= \frac{1}{2} \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}, \\ U_{\sqrt{NOT}} U_{\sqrt{NOT}} &= U_{NOT}. \end{aligned}$$

A Phase gate and a Controlled Phase Shift (CPS) gate are

$$U_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

and

$$U_{CPS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix},$$

respectively. Quantum gates are the basic units of quantum algorithms.

The power of quantum computation is characterized by the quantum parallelism based on superposition and entanglement. If, for example, two qubits are not separable, their state is entangled [60]. The difference between 'unentangled' and 'entangled' states is shown in the following.

$$\begin{aligned} |\psi\rangle_s &= \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \\ &= (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle) \\ &= |\psi\rangle_{q_1} \otimes |\psi\rangle_{q_2} \end{aligned} \tag{2.6}$$

$$\begin{aligned} |\psi\rangle_e &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \\ &\neq |\psi\rangle_{q_1} \otimes |\psi\rangle_{q_2} \end{aligned} \tag{2.7}$$

While the state $|\psi\rangle_s$ of (2.6) is superposed but not entangled, the state $|\psi\rangle_e$ of (2.7) is superposed and entangled. The entangled state cannot be expressed by a tensor product of qubits.

2.3 Summary

In this chapter, evolutionary computation was introduced briefly as a general outline. The history and basics of quantum computation were also described.

3. Quantum-inspired Evolutionary Algorithm (QEA)

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. The representation, the proposed algorithm, and its characteristics are described in the following.

3.1 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 [61]. 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 3.1. A *Q-bit* is defined as the smallest unit of information in QEA, which is defined with a pair of numbers, (α, β), 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 states.

Definition 3.2. A *Q-bit individual* as a string of *m* Q-bits is defined as

$$\left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{array}\right], \tag{3.1}$$

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

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

$$\left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array}\right], \tag{3.2}$$

then the states of the system can be represented as

$$\begin{aligned} &\frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle - \frac{1}{4}|010\rangle - \frac{\sqrt{3}}{4}|011\rangle \\ &+\frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle - \frac{1}{4}|110\rangle - \frac{\sqrt{3}}{4}|111\rangle. \end{aligned} \tag{3.3}$$

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 $\frac{1}{16}$, $\frac{3}{16}$, $\frac{1}{16}$, $\frac{3}{16}$, $\frac{1}{16}$, $\frac{3}{16}$, $\frac{1}{16}$, and $\frac{3}{16}$, respectively. Consequently, the three-Q-bit system of (3.2) contains information of eight states.

Evolutionary algorithm with Q-bit representation has a better characteristic of population diversity than any other representations, since it can represent linear superposition of states probabilistically. Only one Q-bit individual such as (3.2) 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.

3.2 Basic structure of QEA

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

$$\mathbf{q}_j^t = \left[\begin{array}{c|c|c|c} \alpha_{j1}^t & \alpha_{j2}^t & \cdots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \cdots & \beta_{jm}^t \end{array}\right], \tag{3.4}$$

where m is the number of Q-bits, i.e., the string length of the Q-bit individual, and $j = 1, 2, \cdots, n$.

Figure 3.1 and 3.2 show the procedure QEA and the overall structure of QEA that can be explained in the following manner:

i) In the step of 'initialize $Q(t)$,' α_i^0 and β_i^0, $i = 1, 2, \cdots, m$, of all $\mathbf{q}_j^0 = \mathbf{q}_j^t|_{t=0}$, $j = 1, 2, \cdots, n$, are initialized with $\frac{1}{\sqrt{2}}$. It means that one Q-bit individual, $\mathbf{q}_j^0$ represents the linear superposition of all the possible states with the same probability:

$$|\psi_{\mathbf{q}_j^0}\rangle = \sum_{k=1}^{2^m} \frac{1}{\sqrt{2^m}} |X_k\rangle, \tag{3.5}$$

where X_k is the kth state represented by the binary string $(x_1 x_2 \cdots x_m)$, where x_i, $i = 1, 2, \cdots, m$, is either 0 or 1 according to the probability of either $|\alpha_i^0|^2$ or $|\beta_i^0|^2$, respectively. However, it should be noted that the performance of QEA can be influenced by the initial value. The effect of the initial value is discussed in Section 4.1.

ii) This step makes binary solutions in $P(0)$ by observing the states of $Q(0)$, where $P(0) = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \cdots, \mathbf{x}_n^0\}$ at generation $t = 0$. One binary solution $\mathbf{x}_j^0$, $j = 1, 2, \cdots, n$, 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 $|\beta_i^0|^2$, $i = 1, 2, \cdots, m$, of

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 $B(t)$
v) **while** (**not** termination condition) **do**
begin
$t \leftarrow t + 1$
vi) make $P(t)$ by observing the states of $Q(t-1)$
vii) evaluate $P(t)$
viii) update $Q(t)$ using Q-gates
ix) store the best solutions among $B(t-1)$ and $P(t)$ into $B(t)$
x) store the best solution $\mathbf{b}$ among $B(t)$
xi) **if** (global migration condition)
then migrate $\mathbf{b}$ to $B(t)$ globally
xii) **else if** (local migration condition)
then migrate $\mathbf{b}_j^t$ in $B(t)$ to $B(t)$ locally
end
end

Figure 3.1: Procedure QEA.

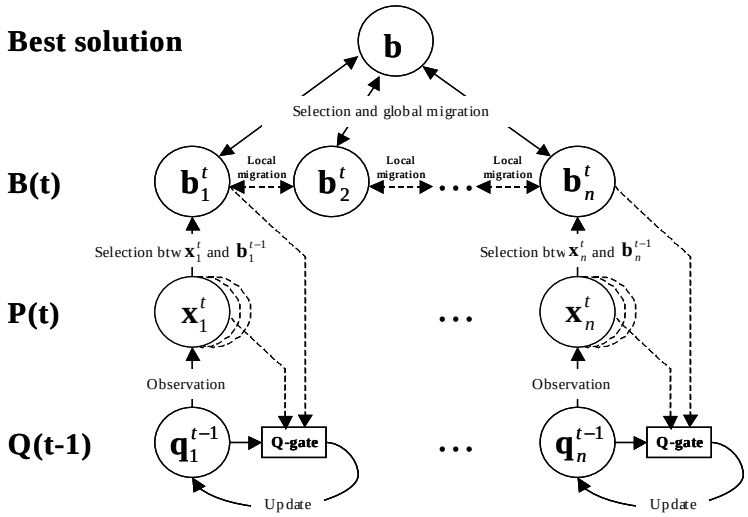


Figure 3.2: Overall structure of QEA.

$\mathbf{q}_j^0$, 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 digital computer, not a quantum computer.

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

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

v) Until the termination condition is satisfied, QEA is running in the **while** loop. In particular, termination criteria are described in Section 3.7.

vi, vii) 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.

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

Definition 3.3. 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 α' and β' are the values of the updated Q-bit.

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

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix}, \tag{3.6}$$

where $\Delta\theta_i$, $i = 1, 2, \cdots, m$, 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. $\Delta\theta_i$ can be obtained as a function of the ith bit of the best solution $\mathbf{b}_j^t$, the ith bit of the binary solution $\mathbf{x}_j^t$, and some meaningful conditions. 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. Hadamard gate is suitable for the algorithms which use the phase information of Q-bit as well as the amplitude information. And it should be noted that H_ϵ gate which is a novel Q-gate as a variation operator is designed in Section 4.2.

ix, x) 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 better fitted than the stored best solution $\mathbf{b}$, the stored solution $\mathbf{b}$ is replaced by the new one.

xi, xii) If the global migration condition is satisfied, the best solution $\mathbf{b}$ is migrated to $B(t)$ globally. If the local migration condition is satisfied, the best one in a local group in $B(t)$ is migrated to others in the same local group. The migration process defined below can induce a variation of the probabilities of a Q-bit individual.

Definition 3.4. 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 all the solutions in the same local group by the best one of them.

Definition 3.5. A *local group* in QEA is defined as the subpopulation affected mutually by a local migration, and its size is the number of individuals in the local group.

3.3 Application example: The knapsack problem

In this section, the detailed algorithm of QEA for the knapsack problem is presented. The knapsack problem (see Appendix A.1) is considered to demonstrate the applicability of QEA to a class of combinatorial optimization problems. For comparison purpose, three types of GA methods are described briefly.

3.3.1 QEA for the knapsack problem

QEA for the knapsack problem consists of a basic structure of QEA and a random repair process to satisfy the capacity constraint. Figure 3.3 shows the algorithm for the knapsack problem.

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 3.2. The ith 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 $\mathbf{x}_j^t$, $j = 1, 2, \cdots, n$, in $P(t)$, a random number r is generated from the range $[0..1]$; 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 jth Q-bit individual. For

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 ($t <$ MAXGEN) **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 (global migration condition)
        then migrate $\mathbf{b}$ to $B(t)$ globally
        else if (local migration condition)
        then migrate $\mathbf{b}_j^t$ in $B(t)$ to $B(t)$ locally
    end
end

Figure 3.3: Procedure QEA for the knapsack problem.

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 shown in Figure 3.4. When the binary string violates the capacity constraint, the random repair method shown in Figure 3.5 is employed, although the greedy repair method guarantees the better solutions.

The **update** procedure of Q-bits is presented in Figure 3.6. A rotation gate $U(\Delta\theta_i)$ is employed to update a Q-bit individual $\mathbf{q}$ as a variation operator. (α_i, β_i)

Procedure make (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

Figure 3.4: Procedure make.

Procedure repair (x)
begin
    knapsack-overfilled $\leftarrow$ false
    if $\sum_{i=1}^{m} w_i x_i > C$
    then knapsack-overfilled $\leftarrow$ true
    while (knapsack-overfilled) **do**
    begin
        select an ith item from the knapsack
        $x_i \leftarrow 0$
        if $\sum_{i=1}^{m} w_i x_i \leq C$
        then knapsack-overfilled $\leftarrow$ false
    end
    while (**not** knapsack-overfilled) **do**
    begin
        select a jth item from the knapsack
        $x_j \leftarrow 1$
        if $\sum_{i=1}^{m} w_i x_i > C$
        then knapsack-overfilled $\leftarrow$ true
    end
    $x_j \leftarrow 0$
end

Figure 3.5: Procedure repair.

Procedure update (q)
begin
    $i \leftarrow 0$
    while ($i < m$) **do**
    begin
        $i \leftarrow i + 1$
        determine $\Delta\theta_i$ with the lookup table
        obtain (α'_i, β'_i) from the following:
            if (**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

Figure 3.6: Procedure update.

of the ith Q-bit is updated as follows:

$$\begin{bmatrix} \alpha'_i \\ \beta'_i \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \tag{3.7}$$

Figure 3.7 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 3.1. Let us define an angle vector $\Theta = [\theta_1 \ \theta_2 \ \cdots \ \theta_8]^T$, where $\theta_1, \ \theta_2, \ \cdots, \ \theta_8$ can be selected easily by intuitive reasoning. For example, if x_i and b_i are 0 and 1, respectively, and if the condition $f(\mathbf{x}) < f(\mathbf{b})$ is true, then:

i) if the Q-bit is located in the first or the third quadrant in Figure 3.7, θ_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

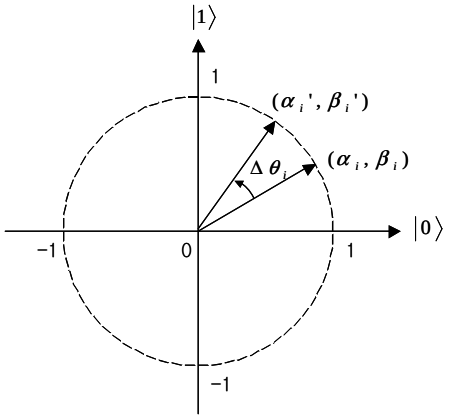


Figure 3.7: Polar plot of the rotation gate for Q-bit individuals.

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 $f(\mathbf{x}) < f(\mathbf{b})$ is true, then:

i) if the Q-bit is located in the first or the third quadrant, θ_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 ranging from 0.001π to 0.1π are recommended for the magnitude of $\Delta\theta_i$, although they depend on the problems. The sign of $\Delta\theta_i$ determines the direction of convergence. The verification of the angle selection is presented in Section 3.4.

x_i	b_i	$f(\mathbf{x}) < f(\mathbf{b})$	$\Delta\theta_i$
0	0	$true$	θ_1
0	0	$false$	θ_2
0	1	$true$	θ_3
0	1	$false$	θ_4
1	0	$true$	θ_5
1	0	$false$	θ_6
1	1	$true$	θ_7
1	1	$false$	θ_8

Table 3.1: Lookup table of $\Delta\theta_i$, where $f(\cdot)$ is the profit, and b_i and x_i are the ith 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.

3.3.2 GA methods for the knapsack problem

There are several GA methods for the knapsack problem [61, 62, 63, 64, 65, 66, 67]. In this section, 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 [68].

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 [69, 70]. Two types of penalties are considered, such as logarithmic penalty and linear penalty:

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

where ρ 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:

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, and 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 ith 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.

3.3.3 Experimental results

In all experiments, strongly correlated sets of data were considered:

$$\begin{aligned} w_i &= uniformly\ random[1, 10], \\ p_i &= w_i + 5. \end{aligned}$$

The average knapsack capacity (see Appendix A.1) was used for the knapsack constraint. Three knapsack problems with 100, 250, and 500 items were considered, and the data were unsorted.

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, and the local group size was 2. Figure 3.8 shows the results of QEA1, QEA2, and QEA3 on the knapsack problems with 100, 250 and 500 items for finding good parameter settings of θ_3 and θ_5 of the lookup table. The values of 0.0025π, 0.005π, 0.01π, 0.02π, and 0.05π were tested for θ_3 and $-\theta_5$. All the best profits were averaged over 30 runs, and the maximum number of generations was 1,000. It should be noted that the results of the cases with the same value of θ_3 and $-\theta_5$ were better than the others. From the results, the values of 0.01π and -0.01π were selected for θ_3 and θ_5, respectively.

The population sizes of conventional GAs (CGAs) were 1 and 10. To discover good parameter settings of CGAs, the values of 0.001, 0.01, 0.05, and 0.1 for mutation and of 0.01, 0.05, 0.1, 0.3, 0.5, and 0.7 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). Figure 3.9 shows the results of six CGAs on the knapsack problems with 100, 250 and 500 items to find good parameter settings. All the best

		CGAs		QEAs		
		$Rep2$ (1)	$Rep2$ (10)	$QEA1$ (1)	$QEA2$ (10)	$QEA3$ (10)
100	b.	592.4	607.7	597.7	612.7	612.7
	m.	576.4	599.2	591.8	606.3	609.5
	w.	557.2	587.6	582.5	597.7	607.6
	σ	7.975	4.673	4.840	3.308	2.404
	t_e	0.015	0.127	0.021	0.199	0.203
250	b.	1444.9	1479.8	1480.2	1515.2	1525.2
	m.	1415.1	1462.5	1464.5	1508.1	1518.7
	w.	1394.2	1440.2	1445.1	1495.2	1515.2
	σ	12.480	8.788	9.554	5.427	2.910
	t_e	0.035	0.308	0.055	0.531	0.558
500	b.	2820.0	2895.4	2899.7	3004.6	3025.8
	m.	2772.9	2864.5	2876.4	2980.8	3008.0
	w.	2712.3	2841.1	2836.2	2966.3	2996.1
	σ	21.453	15.257	12.832	9.411	8.039
	t_e	0.068	0.615	0.117	1.212	1.258

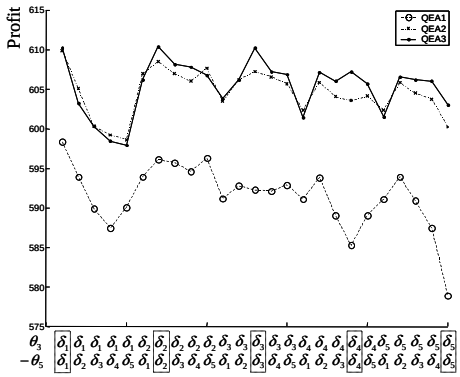
Table 3.2: Experimental results of the knapsack problem. The number of items 100, 250 and 500, the maximum number of generations 1,000, the number of runs 30. The parenthesized values are the population sizes. $Rep2$ means the algorithm implemented by the greedy repair method, and $b.$, $m.$, and $w.$ mean $best$, $mean$, and $worst$, respectively. σ and t_e(sec/run) represent the standard deviation and the elapsed time per run, respectively.

profits were averaged over 30 runs, and the maximum number of generations was 1,000. From Figure 3.9, we could select $(0.05, 0.1)$ for the population size 1 and $(0.01, 0.7)$ for the population size 10 as ordered pairs of mutation and crossover probabilities that gave the maximum profit.

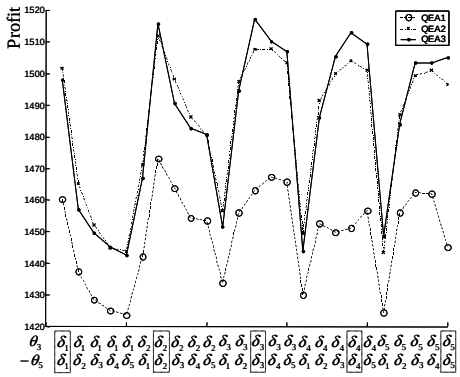
As a performance measure of the algorithms, we collected the best solution found within 1,000 generations over 30 runs, and we checked the elapsed time per run, which are summarized in Table 3.2, where only $Rep2$ among CGAs is shown because it outperformed all other CGAs. As Table 3.2 shows, QEAs yielded much better results compared to $Rep2$, except in the results of $Rep2$ (10) and $QEA1$ with

100 items, which, however, 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 better solutions within a short span of time as compared to CGAs'.

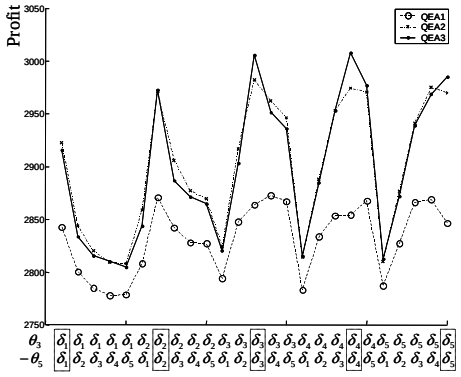
Figure 3.10 shows the progress of the mean of best profits and the mean of average profits of population found by QEA2, QEA3 and CGA (Rep2) over 30 runs for 100, 250 and 500 items. QEAs performed a lot better than CGA in terms of convergence rate and profit amount. QEAs showed a faster convergence rate than CGA. QEAs' final profits were much larger than CGA's in 1,000 generations. The tendency of convergence is shown clearly in the results of the mean of average profits for the population. In the beginning, the convergence rates of all the algorithms increased. However, the convergence rate of CGA decreased gradually due to its premature convergence. As shown in Figure 3.10 (d) and (f), QEAs displayed no premature convergence in average profits throughout the 1,000 generations. In particular, the results on QEA2 and QEA3 should be mentioned that initially, QEA3 showed a slower convergence rate than QEA2. However, QEA3 outperformed QEA2 in profits after about 500 generations, since QEA3, with a global migration process every 100 generations and a local migration process in every generation, can increase the population diversity.



(a) 100 items

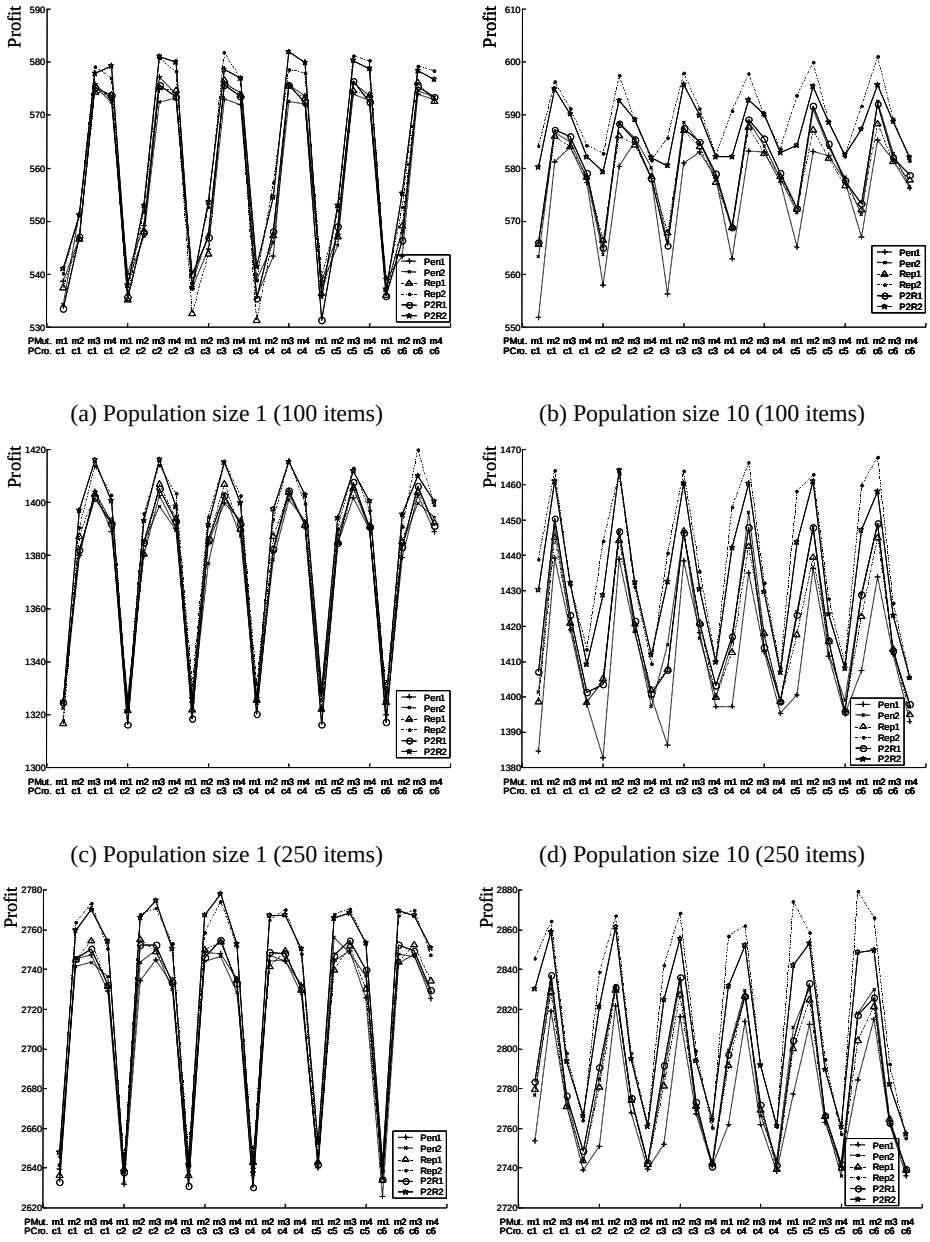


(b) 250 items



(c) 500 items

Figure 3.8: Best profits of QEAs on the knapsack problems with 100, 250 and 500 items to find good parameter settings of θ_3 and θ_5 of Table 3.1. The vertical axis is the best profit averaged over 30 runs, and the horizontal axis is the parameter settings of ordered pairs of θ_3 and $-\theta_5$. δ_1, δ_2, δ_3, δ_4, and δ_5 are 0.0025π, 0.005π, 0.01π, 0.02π, and 0.05π, respectively.



(a) Population size 1 (100 items)

(b) Population size 10 (100 items)

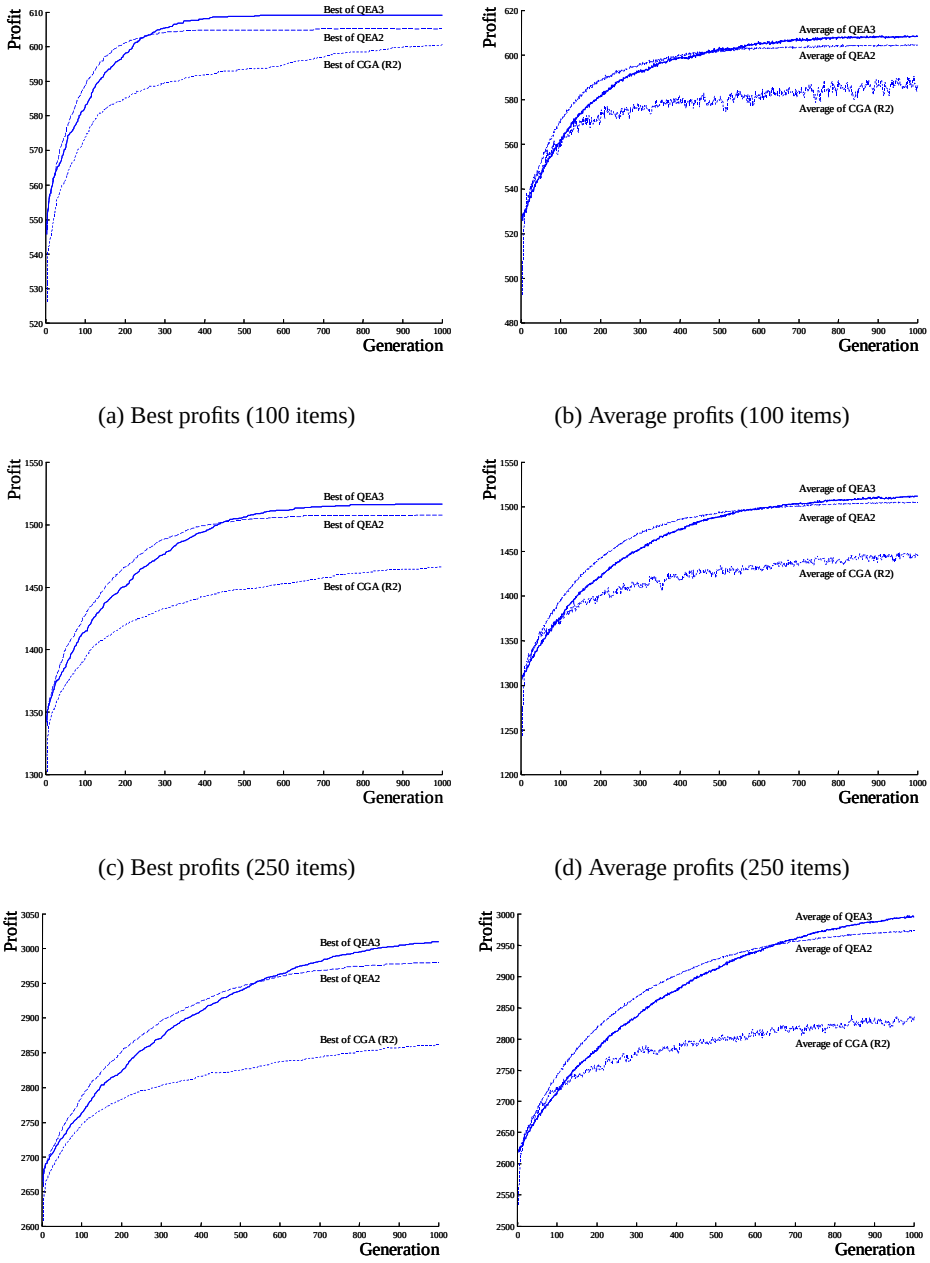
(c) Population size 1 (250 items)

(d) Population size 10 (250 items)

(e) Population size 1 (500 items)

(f) Population size 10 (500 items)

Figure 3.9: 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 to m4 are 0.001, 0.01, 0.05, and 0.1, and c1-c6 are 0.01, 0.05, 0.1, 0.3, 0.5, and 0.7.



(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)

Figure 3.10: Comparison of QEAs and CGA on the knapsack problem. The CGA is $Rep2$ 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.

3.4 Verification of the angle selection

In this section, the selection of the angle parameters for the rotation gate is verified. In Section 3.3, it was suggested to set a positive number for θ_3, a negative number for θ_5, and 0 for the rest of the angle parameters in Θ of Table 3.1 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 1,000. The values of 0, 0.005π, and -0.005π were used for each of the eight angle parameters. That is, 3^8 cases of Θ were tried. Figure 3.11 shows the experimental results carried out step by step to find proper signs $(0, +, -)$ of the angle parameters: (a) 3^8 cases of Θ, (b) 3^7 cases of Θ when θ_1 was selected as 0, (c) 3^6 cases of Θ when both of θ_1 and θ_2 were selected as 0, (d) 3^5 cases of Θ when both of θ_1 and θ_2 were 0, and θ_3 was selected as a positive number, 0.005π, (e) 3^4 cases of Θ when both of θ_1 and θ_2 were 0, θ_3 was 0.005π, and θ_4 was selected as 0, and (f) 3^3 cases of Θ when both of θ_1 and θ_2 were 0, θ_3 was 0.005π, θ_4 was 0, and θ_5 was selected as a negative number, -0.005π. The results on θ_2, θ_4, θ_6, and θ_8, that is, the cases in which $f(\mathbf{x}) < f(\mathbf{b})$ is false are worthwhile to be mentioned that the values of θ_2, θ_4, θ_6, and θ_8 had little effect on the results as shown in Figures 3.11(b), (d), and (f), respectively. It means that θ_2, θ_4, θ_6, and θ_8 can be set to any one among 0, 0.005π, and -0.005π. In the results on the cases in which $f(\mathbf{x}) < f(\mathbf{b})$ is true, the values of 0, 0.005π, -0.005π, and 0 for θ_1, θ_3, θ_5, and θ_7, respectively, made better solutions. From these experimental results, Θ 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 consistent with the intuitive reasoning given in Section 3.3.

Three numerical problems are considered to show that the results on Θ can be applied to other optimization problems. To deal with numerical problems, real values of the variables should be encoded as binary strings since QEA uses a Q-bit

representation to generate a binary bit. The three numerical problems are as follows:

Problem 1: Maximize $f_1(\mathbf{x}) = 100-(100(x_1^2-x_2)^2+(1-x_1)^2)$, where $-2.048 \leq x_i \leq 2.048$. The global maximum value is 100 at $(x_1, x_2) = (1, 1)$. This function is a modified version of De Jong function (1) (see Appendix A.2).

Problem 2: Maximize $f_2(\mathbf{x}) = -\sum_{i=1}^{5} integer(x_i)$, where $-5.12 \leq x_i \leq 5.12$. The global maximum value is 30 for all $-5.12 \leq x_i < -5.0$. This function is a modified version of De Jong function (2).

Problem 3: Maximize $f_3(\mathbf{x}) = 100.98 - \frac{1}{\frac{1}{K}+\sum_{j=1}^{25} g_j^{-1}(x_1,x_2)}$, where $g_j(x_1, x_2) = c_j + \sum_{i=1}^{2}(x_i - a_{ij})^6$, where $-65.536 \leq x_i \leq 65.536$, $K = 500$, $c_j = j$, and

$$[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32 \end{bmatrix}.$$

The global maximum value is 100 at $(x_1, x_2) = (-32, -32)$. This function is a modified version of De Jong function (3).

Each variable was encoded as a 25-bit string. The population size was 1. The maximum number of generations was $1,000$. The values of 0, 0.005π, and -0.005π were used for each of the eight angle parameters. Figures 3.12, 3.13 and 3.14 show the results of Problems 1, 2, and 3, respectively, carried out step by step to find proper signs $(0, +, -)$ of the angle parameters. The results on θ_2, θ_4, θ_6, and θ_8, that is, the cases which $f(\mathbf{x}) < f(\mathbf{b})$ is false, are worthwhile to be mentioned that the values of θ_2, θ_4, θ_6, and θ_8 had little effect on the results as shown in (b), (d), and (f) of Figures 3.12, 3.13 and 3.14. These are the same results of the knapsack problem as shown in Figure 3.11. The set of Θ for finding the maximum value of each problem was obtained from the results as follows:

f_1: $[0 * p * 0 * n *]^T$, $[0 * p * 0 * 0 *]^T$, $[0 * p * n * 0 *]^T$, $[p * p * n * 0 *]^T$, and $[p * p * n * n *]^T$;

f_2: $[0 * p * n * 0 *]^T$, $[0 * p * n * n *]^T$, $[0 * n * n * 0 *]^T$, $[0 * n * n * n *]^T$,

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
0	0.62	$*$	0.08	$*$	0.13	$*$	0.53	$*$
p	0.3	$*$	0.75	$*$	0	$*$	0	$*$
n	0.08	$*$	0.17	$*$	0.87	$*$	0.47	$*$

Table 3.3: Average frequencies of 0, p, and n for each θ_i in Θ from Problems 1, 2, and 3. The values are scaled between 0 and 1.

	x_i	b_i	$\Delta\theta_i$	$rec.$
$f(\mathbf{x}) < f(\mathbf{b})$ $true$	0	0	θ_1	0
	0	1	θ_3	p
	1	0	θ_5	n
	1	1	θ_7	0

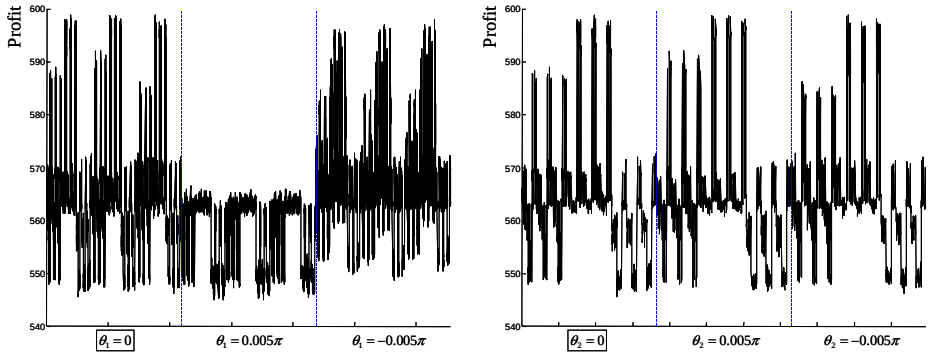
Table 3.4: Simplified lookup table of $\Delta\theta_i$, where b_i and x_i are the ith bits of the best solution $\mathbf{b}$ and the binary solution $\mathbf{x}$, respectively. $rec.$ means the recommended value of $\Delta\theta_i$. p is a positive number, and n is a negative number.

$[0*0*n*0*]^T$, $[0*0*n*n*]^T$, $[n*n*n*0*]^T$, and $[n*n*n*n*]^T$;

f_3: $[0*p*n*0*]^T$, $[0*p*n*n*]^T$, $[p*p*n*0*]^T$, and $[p*p*n*n*]^T$.

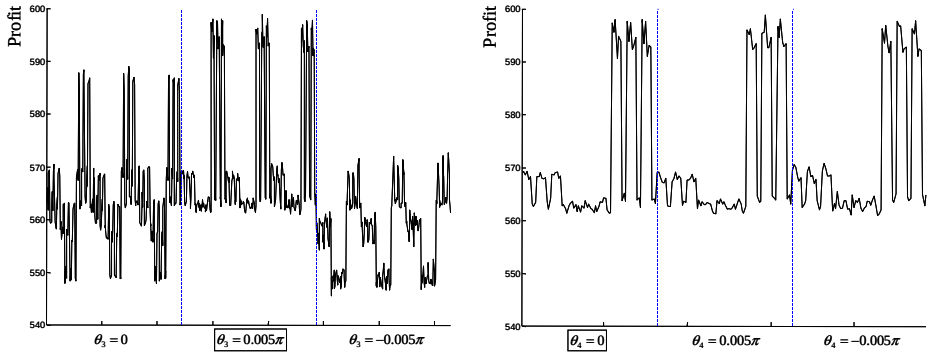
Table 3.3 shows the average frequencies of 0, p, and n for each θ_i in Θ from the above results. From the table, $[0 * p * n * 0*]^T$ has a higher frequency and is included in each set of Θ for Problems 1, 2, and 3. It means that Θ can be assigned as $[0 * p * n * 0*]^T$ for other problems.

From the empirical results, Table 3.1 for the rotation gate can be simplified as Table 3.4. It should be noted that θ_1 and θ_7 can be assigned a nonzero value in compliance with the application problems.



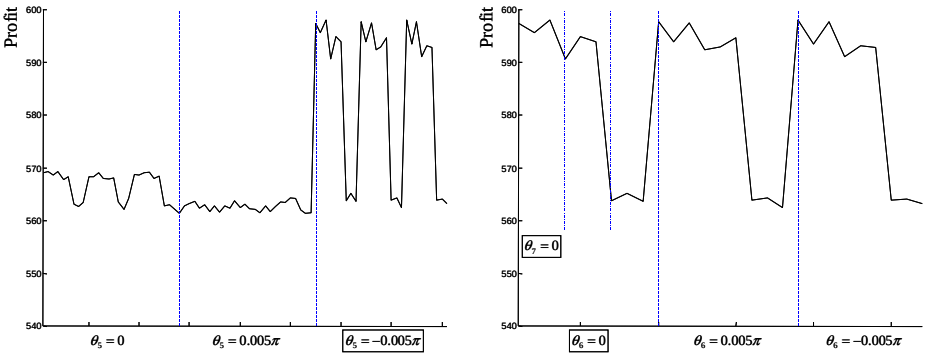
(a) $\Theta = [* \ * \ * \ * \ * \ * \ *]^T$

(b) $\Theta = [0 \ * \ * \ * \ * \ * \ *]^T$



(c) $\Theta = [0 \, 0 \ * \ * \ * \ * \ *]^T$

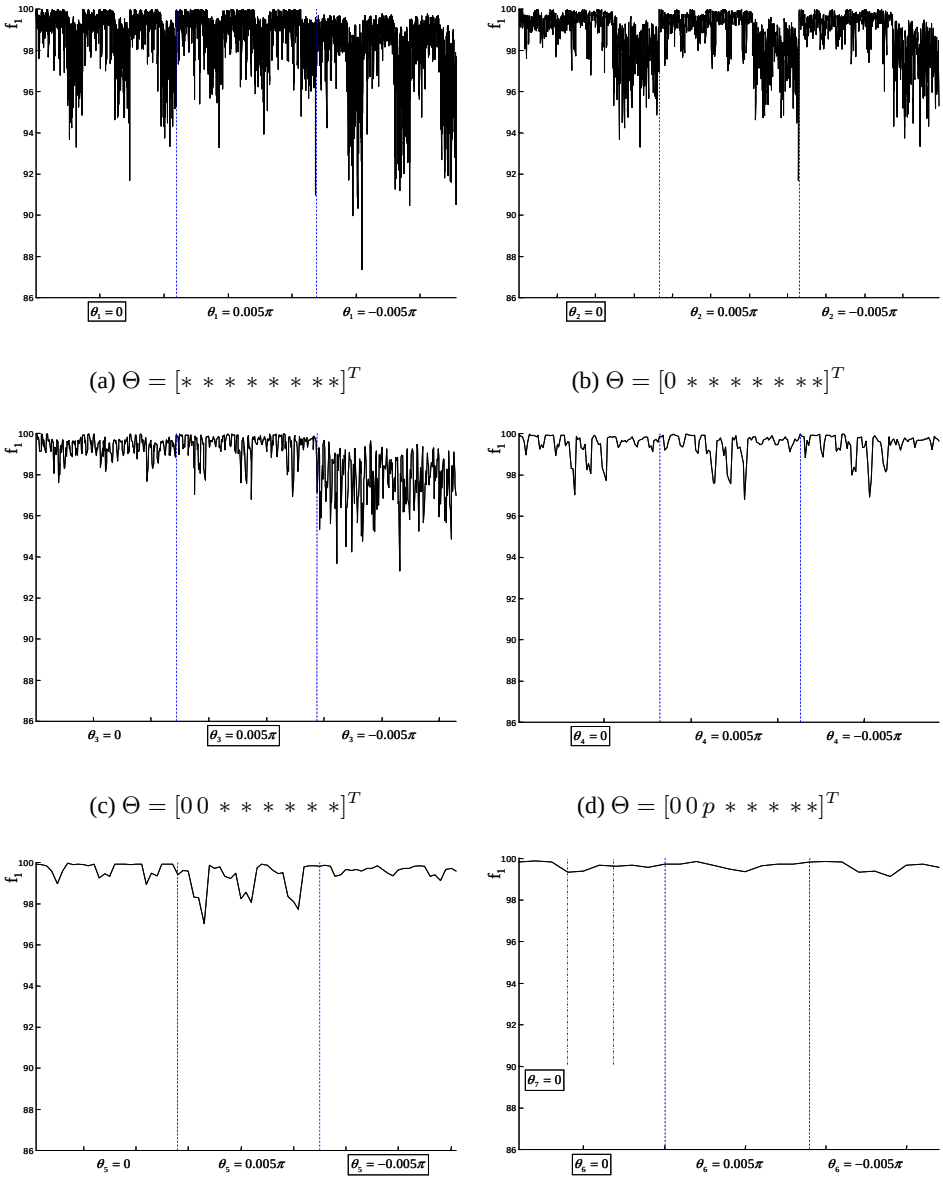
(d) $\Theta = [0 \, 0 \, p \ * \ * \ * \ *]^T$



(e) $\Theta = [0 \, 0 \, p \, 0 \ * \ * \ * \ *]^T$

(f) $\Theta = [0 \, 0 \, p \, 0 \, n \ * \ * \ *]^T$

Figure 3.11: Best profits of QEA1 on the knapsack problem with 100 items to find proper signs of the angle parameters of Table 3.1. 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, 0.005π, and -0.005π. p and n were set to 0.005π and -0.005π, respectively.



(a) $\Theta = [* \ * \ * \ * \ * \ * \ * \ *]^T$

(b) $\Theta = [0 \ * \ * \ * \ * \ * \ * \ *]^T$

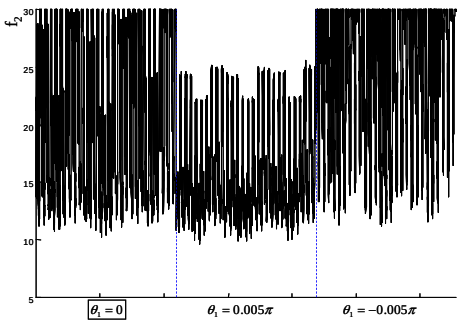
(c) $\Theta = [0 \, 0 \ * \ * \ * \ * \ * \ *]^T$

(d) $\Theta = [0 \, 0 \, p \ * \ * \ * \ * \ *]^T$

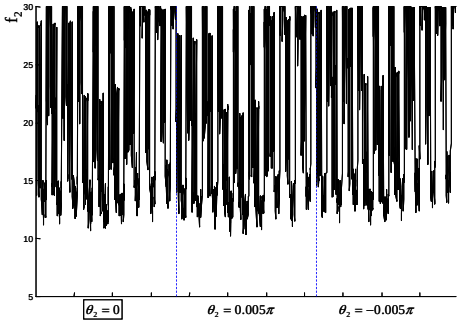
(e) $\Theta = [0 \, 0 \, p \, 0 \ * \ * \ * \ *]^T$

(f) $\Theta = [0 \, 0 \, p \, 0 \, n \ * \ * \ *]^T$

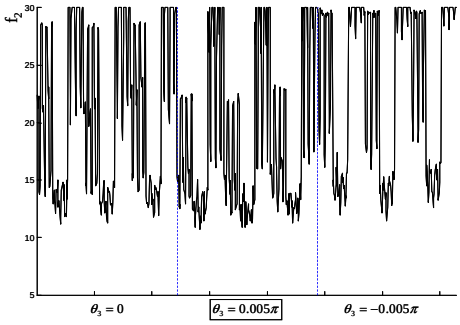
Figure 3.12: Results of Problem 1 to find proper signs of the angle parameters of Table 3.1. The vertical axis is the function value of $f_1(\mathbf{x})$ averaged over 30 runs, and the horizontal axis is the parameter settings of the angle values. $*$ could be set to 0, 0.005π, and -0.005π. p and n were set to 0.005π and -0.005π, respectively.



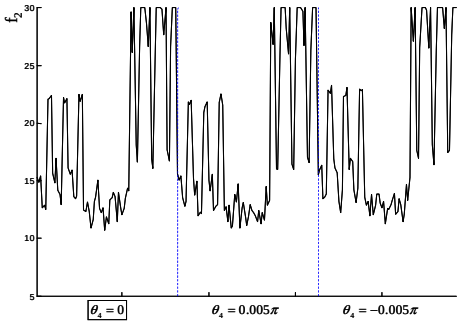
(a) $\Theta = [* \ * \ * \ * \ * \ * \ * \ *]^T$



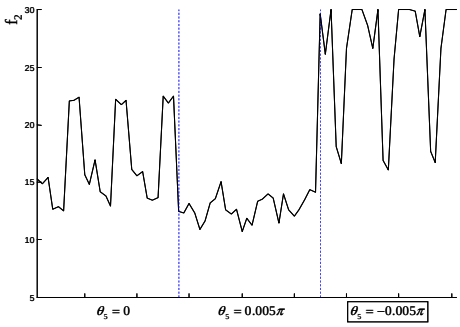
(b) $\Theta = [0 \ * \ * \ * \ * \ * \ *]^T$



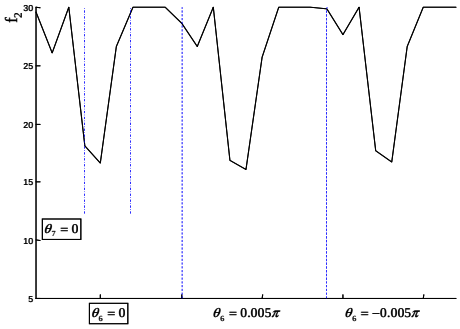
(c) $\Theta = [0 \, 0 \ * \ * \ * \ * \ *]^T$



(d) $\Theta = [0 \, 0 \, p \ * \ * \ * \ *]^T$

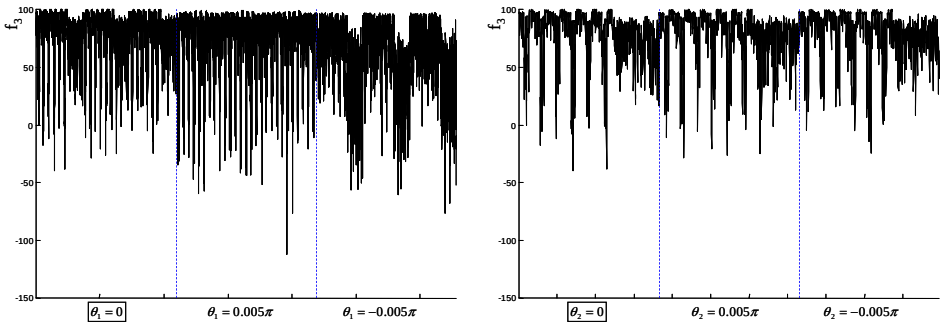


(e) $\Theta = [0 \, 0 \, p \, 0 \ * \ * \ * \ *]^T$



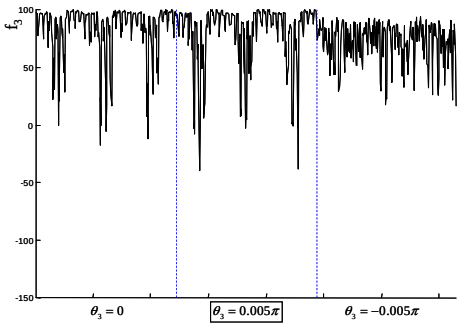
(f) $\Theta = [0 \, 0 \, p \, 0 \, n \ * \ *]^T$

Figure 3.13: Results of Problem 2 to find proper signs of the angle parameters of Table 3.1. The vertical axis is the function value of $f_2(\mathbf{x})$ averaged over 30 runs, and the horizontal axis is the parameter settings of the angle values. $*$ could be set to 0, 0.005π, and -0.005π. p and n were set to 0.005π and -0.005π, respectively.

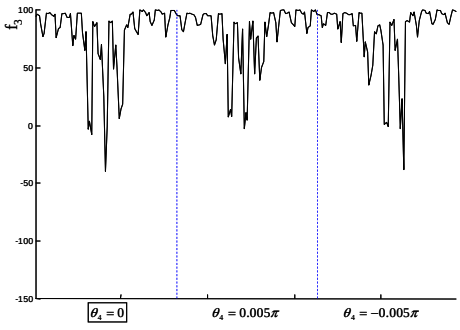


(a) $\Theta = [* \ * \ * \ * \ * \ * \ * \ *]^T$

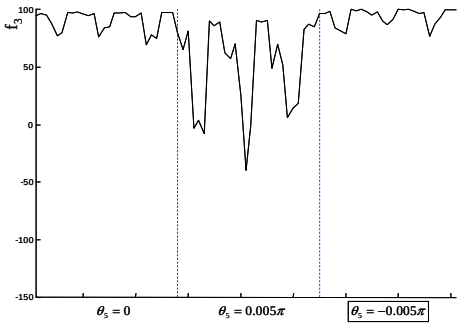
(b) $\Theta = [0 \ * \ * \ * \ * \ * \ *]^T$



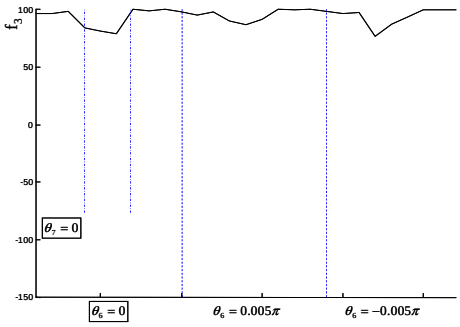
(c) $\Theta = [0 \, 0 \ * \ * \ * \ * \ *]^T$



(d) $\Theta = [0 \, 0 \, p \ * \ * \ * \ *]^T$



(e) $\Theta = [0 \, 0 \, p \, 0 \ * \ * \ * \ *]^T$



(f) $\Theta = [0 \, 0 \, p \, 0 \, n \ * \ * \ *]^T$

Figure 3.14: Results of Problem 3 to find proper signs of the angle parameters of Table 3.1. The vertical axis is the function value of $f_3(\mathbf{x})$ averaged over 30 runs, and the horizontal axis is the parameter settings of the angle values. $*$ could be set to 0, 0.005π, and -0.005π. p and n were set to 0.005π and -0.005π, respectively.

3.5 Investigation of the characteristics

In this section, the characteristic of the proposed QEA is investigated. 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 3.3. 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 as shown in Figure 3.15. In this problem, the best profit satisfying the capacity constraint was 62.192938 at the 127th. The solutions with larger profit 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 3.3. Figure 3.16 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 (3.5), we have a probability of 0.001 ($\frac{1}{\sqrt{2^{10}}}^2 = \frac{1}{2^{10}}$) for each solution which is shown in Figure 3.16 (a), (b), and (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 Figure 3.15. 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 to 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.

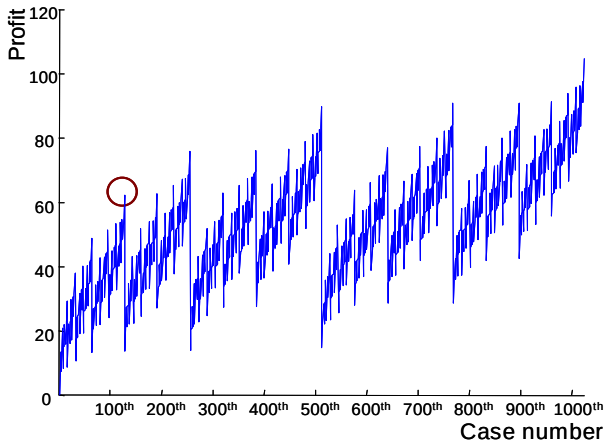
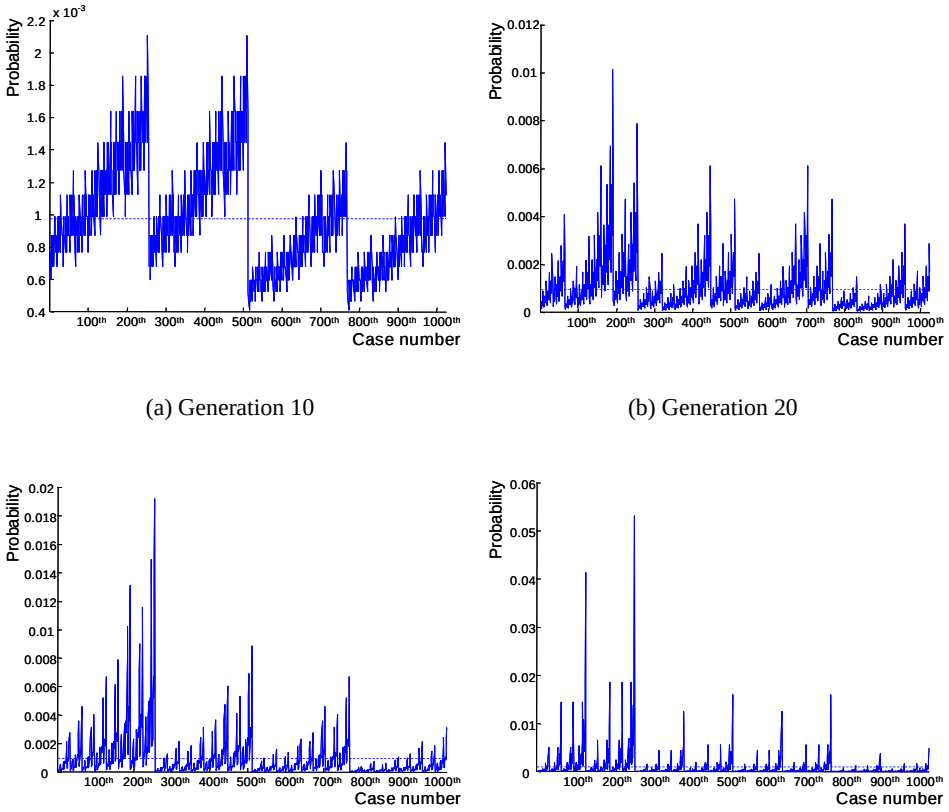


Figure 3.15: Profit values of 1024 cases in the knapsack problem with ten items obtained by a simple calculation. The vertical axis is the profit values of the knapsack, and the horizontal axis is the number of 1024 cases selected as a subset from ten items. The best profit satisfying the capacity constraint is marked with O.

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 Figure 3.15. As the probabilities of the solutions with larger profit increase, QEA starts a local search. Finally, the probability of the best solution 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.



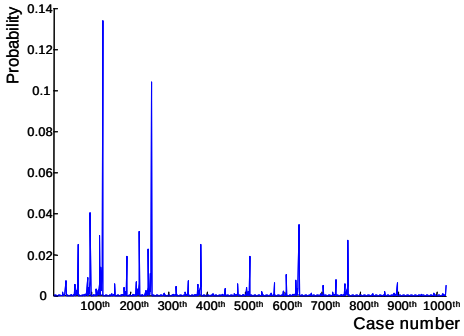
(a) Generation 10

(b) Generation 20

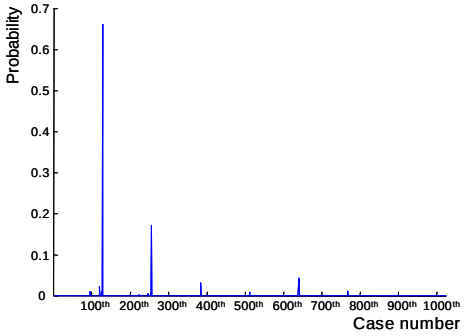
(c) Generation 30

(d) Generation 40

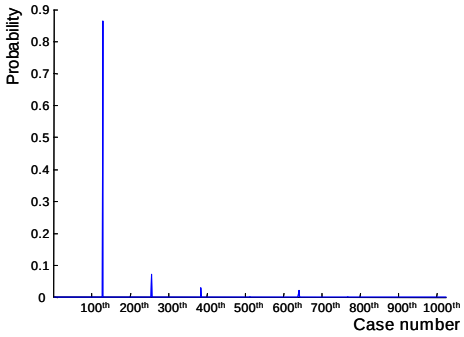
Figure 3.16: Probabilities of all solutions using a Q-bit individual. The vertical axis is the probability of the solution, and the horizontal axis is the number of 1024 cases selected as a subset from ten items.



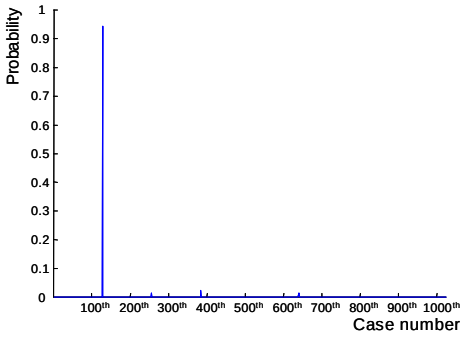
(e) Generation 50



(f) Generation 100



(g) Generation 200



(h) Generation 300

Figure 3.16: *(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.

3.6 Verification of the QEA algorithm

There have been some works done based on the theoretical analysis of EAs for certain simple functions [71, 72, 73, 74, 75, 76]. However, the theories behind these analyses cannot be applied to the analysis of QEA, since the structure of the QEA algorithm is quite different from any other EAs. In this section, the reason why and how QEA works is investigated by using a simple function with two viewpoints like its exploitation and exploration.

3.6.1 Exploitation

A theoretical model for the whole process of the QEA algorithm is hard to find, since each state of QEA is dependent on the past history. However, if a simplified model for a segment of the QEA process (as shown in Figure 3.17) is considered, the abstract model can be regarded as a Markov chain.

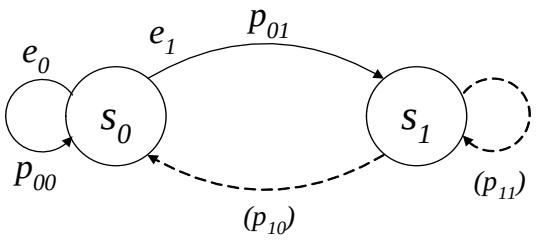


Figure 3.17: Simplified process model for a segment of the QEA process.

The simplified model of the segment process of QEA represents the process which is defined during the state holding period t_h between the t_sth generation when the current best solution is visited and the t_eth (or $(t_s + t_h)$th) generation when the current best solution jumps to another better solution. In Figure 3.17, s_0 is the state which indicates the state when the current best solution is maintained, and s_1 is the state which indicates the state when the current best solution is changed to another better solution. e_0 is the event that states that the observed solution is worse than

the current best solution, and e_1 is the event that states that the observed solution is better than the current best solution. And p_{ij}, $i, j = 0, 1$, is the transition probability from state i to state j. It should be noted that p_{10} and p_{11} are not needed, since the process corresponding to this model is terminated if the state is changed from s_0 to s_1. The whole process of QEA can be regarded as a sequence of segment processes.

The segment process of QEA (SPQEA) is described by using Markov process [77] as follows:

$$
\begin{aligned}
&SPQEA = (\mathbb{E}, \mathbb{S}, \Gamma, p, p_0) \\
&\mathbb{E} = \{e_0, e_1\}, \;\; \mathbb{S} = \{s_0, s_1\}, \\
&\Gamma(s_0) = \{e_0, e_1\}, \;\; \Gamma(s_1) = \{\}, \\
&p(s_0; s_0, e_0) = p_{00}, \;\; p(s_1; s_0, e_1) = p_{01}, \\
&p_0(s_0) = 1, \;\; p_0(s_1) = 0,
\end{aligned}
\tag{3.8}
$$

where $\mathbb{E}$ is a event set, $\mathbb{S}$ a state space, $\Gamma(s)$ a set of feasible events defined for all $s \in \mathbb{S}$ with $\Gamma(s) \subseteq \mathbb{E}$, $p(s'; s, e')$ a state transition probability defined for all $s, s' \in \mathbb{S}$, $e' \in \mathbb{E}$, and such that $p(s'; s, e') = 0$ for all $e' \notin \Gamma(s)$, and $p_0(s)$ the probability mass function $P[S_0 = s]$, $s \in \mathbb{S}$, of the initial state S_0.

Let us consider the ONEMAX problem as follows:

ONEMAX problem: Maximize

$$
\text{ONEMAX}(\mathbf{x}) = \sum_{i=1}^{m} x_i, \tag{3.9}
$$

where x_i is the ith bit of $\mathbf{x}$, m is the length of $\mathbf{x}$, and the global maximum value is m at $\mathbf{x} = 111 \cdots 1$.

Let us suppose that all the rotation angles of the rotation gate in QEA are zeros. Then the QEA process is the same as the process of random search. In this case,

each solution in the search space has the same probability and its probability is invariant all the time. It means that this process can be modelled by using only one SPQEA with $p_{00} = \frac{2^m-1}{2^m}$ and $p_{01} = \frac{1}{2^m}$. The expected running number of generations for this model is described in the following.

Theorem 3.1. *The expected running number of generations t_h of the random search is*

$$t_h = -\frac{\log 2}{\log(1-p_{01})}, \tag{3.10}$$

where p_{01} is the transition probability from state s_0 to state s_1.

Proof. Let $V(s)$ be the number of generations spent at state s when it is visited.

$$\begin{aligned}
P[V(s_0)=1] &= p_{01} \\
P[V(s_0)=2] &= p_{00}p_{01} = (1-p_{01})p_{01} \\
P[V(s_0)=3] &= p_{00}^2 p_{01} = (1-p_{01})^2 p_{01} \\
&\vdots = \vdots \\
P[V(s_0)=t] &= p_{00}^{t-1} p_{01} = (1-p_{01})^{t-1} p_{01}
\end{aligned}$$

To give the expected running number of generations t_h, the summation of the probability $P[V(s_0)=k]$ from $k=1$ to $k=t_h$ should be $\frac{1}{2}$.

$$\begin{aligned}
\sum_{t=1}^{t_h} P[V(s_0)=t] &= 1-(1-p_{01})^{t_h} = \frac{1}{2} \\
\therefore t_h &= -\frac{\log 2}{\log(1-p_{01})}.
\end{aligned}$$

Theorem 3.2. *The expected running number of generations t_h of the random search*

for the ONEMAX *problem for length* m *is*

$$t_h = -\frac{\log 2}{\log(1-\frac{1}{2^m})}. \tag{3.11}$$

Proof. Each solution in the search space for the random search has the same probability $\frac{1}{2^m}$ and its probability is invariant all the time. Let s_0 be the state when the current best solution is one of all the possible solutions except the global maximum. Then the transition probabilities p_{00} and p_{01} are $\frac{2^m-1}{2^m}$ and $\frac{1}{2^m}$, respectively. By Theorem 3.1,

$$t_h \;=\; -\frac{\log 2}{\log(1-p_{01})} \;=\; -\frac{\log 2}{\log(1-\frac{1}{2^m})}.$$

However, if the rotation angles are not zeros, the QEA process should be considered as a sequence of SPQEA models. Also, one SPQEA model should not be considered as a homogeneous Markov chain, since the transition probability p_{ij} is dependent on generation t. Let us consider only one segment of the QEA process, SPQEA. The transition probability at the generation t is supposed to be $p_{01}(t) = \xi(t)p_{01}(t-1)$, where $\xi(t)$ is the increasing rate of the transition probability $p_{01}(t)$, $0 < p_{01}(t) \leq 1$, $\xi(1) = 1$, and $1 < \xi(t) \ll \frac{1}{p_{01}(t)}$ for $t > 1$, the expected running number of generations of SPQEA can be stated as follows.

Theorem 3.3. *The expected running number of generations* t_h *of SPQEA with time-varying transition probability can be approximated as*

$$t_h \;\approx\; \frac{\log\left(1-\frac{\xi}{2}+\frac{\xi-1}{2p_{01}(0)}\right)}{\log(\xi-\xi p_{01}(0))}, \tag{3.12}$$

where $p_{01}(0)$ *is the initial transition probability from state* s_0 *to state* s_1 *and* ξ *is a constant satisfying* $\sum_{k=1}^{t_h} p_{01}(k) = \sum_{k=1}^{t_h} \xi^{k-1} p_{01}(0)$.

Proof. Let $p_{01}(t)$ be the transition probability from s_0 to s_1 and $\xi(t)$ the increasing

rate of the transition probability at the generation t, where $\xi(t) = \frac{p_{01}(t)}{p_{01}(t-1)}$ for $t > 1$ and $\xi(1) = 1$. The probabilities for the state holding period of s_0 are

$$\begin{aligned}
P[V(s_0) = 1] &= p_{01}(1) = \xi(1)p_{01}(0) = p_{01}(0) \\
P[V(s_0) = 2] &= (1 - p_{01}(1))\, p_{01}(2) = (1 - p_{01}(0))\, \xi(2)p_{01}(0) \\
P[V(s_0) = 3] &= (1 - p_{01}(0))(1 - \xi(2)p_{01}(0))\, \xi(3)\xi(2)p_{01}(0) \\
P[V(s_0) = 4] &= (1 - p_{01}(0))(1 - \xi(2)p_{01}(0))(1 - \xi(3)\xi(2)p_{01}(0)) \times \\
&\quad \xi(4)\xi(3)\xi(2)p_{01}(0) \\
&\vdots = \vdots .
\end{aligned}$$

Let $\xi(t)$ be a constant ξ satisfying $\sum_{k=1}^{t} p_{01}(k) = \sum_{k=1}^{t} \xi^{k-1}p_{01}(0)$, the above can be rewritten as

$$\begin{aligned}
P[V(s_0) = 1] &= p_{01}(0) \\
P[V(s_0) = 2] &= (1 - p_{01}(0))\, \xi p_{01}(0) \\
P[V(s_0) = 3] &= (1 - p_{01}(0))(1 - \xi p_{01}(0))\, \xi^2 p_{01}(0) \\
P[V(s_0) = 4] &= (1 - p_{01}(0))(1 - \xi p_{01}(0))(1 - \xi^2 p_{01}(0))\, \xi^3 p_{01}(0) \\
&\vdots = \vdots \\
P[V(s_0) = t] &= \prod_{k=0}^{t-2} (1 - \xi^k p_{01}(0))\, \xi^{t-1} p_{01}(0).
\end{aligned}$$

Since ξ can be considered as $\left(1 + \frac{\delta}{p_{01}(0)}\right)$, where $0 < \delta \ll p_{01}(0)$, $P[V(s_0) = t]$ can be approximated as

$$P[V(s_0) = t] \approx (1 - p_{01}(0))^{t-1}\, \xi^{t-1} p_{01}(0).$$

To give the expected running number of generations t_h, the summation of the probability $P[V(s_0) = k]$ from $k = 1$ to $k = t_h$ should be $\frac{1}{2}$. Therefore, the expected

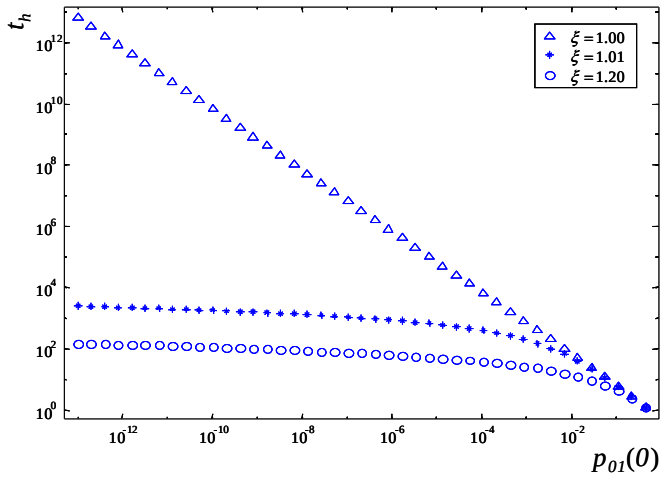


Figure 3.18: Comparison of the expected running number of generations (t_h) with respect to the initial transition probability ($p_{01}(0)$) between QEA ($\xi = 1.01$ and 1.2) and random search ($\xi = 1.0$). ξ is the increasing rate of (3.12). A logarithmic (base 10) scale is used for the horizontal and vertical axes.

running number of generations of SPQEA is obtained as

$$\begin{aligned} \sum_{t=1}^{t_h} P[V(s_0) = t] &\approx p_{01}(0)\frac{1-(1-p_{01})^{t_h}\xi^{t_h}}{1-(1-p_{01})\xi} = \frac{1}{2} \\ \therefore t_h &\approx \frac{\log\left(1-\frac{\xi}{2}+\frac{\xi-1}{2p_{01}(0)}\right)}{\log(\xi-\xi p_{01}(0))}. \end{aligned}$$

It should be noted that if ξ is 1, t_h of (3.12) remains same as that of (3.10) for random search. Figure 3.18 shows the expected running number of generations t_h with respect to the initial transition probability $p_{01}(0)$. In the case of random search ($\xi = 1.0$), if $p_{01}(0)$ is small, t_h is very large, e.g. $t_h|_{p_{01}(0)=1.0\times10^{-13}} = 6.9 \times 10^{12}$ at $\xi = 1.0$. However, for the cases of QEA ($\xi = 1.01$ and 1.2), the expected running number of generations is much smaller than that of random search, e.g. $t_h|_{p_{01}(0)=1.0\times10^{-13}} = 2,475$ at $\xi = 1.01$ and 151 at $\xi = 1.2$.

Let us consider the ONEMAX problem for length m, where $m = 4$. If the initial state s_0 has the current best solution of 1100, the transition probability is

$t=1$			$t=2$			$t=3$		
$\mathbf{x}_0$	$p_{01}(1)$	$\xi(1)$	$\mathbf{x}_1$	$p_{01}(2)$	$\xi(2)$	$\mathbf{x}_2$	$p_{01}(3)$	$\xi(3)$
1100	0.3125	1.0	0000	0.3849	1.2318	0000	0.4590	1.1923
						0001	0.4168	1.0828
						0010	0.4168	1.0828
						0100	0.4209	1.0935
						1000	0.4209	1.0935
			0001	0.3458	1.1067	0000	0.4168	1.2052
						0001	0.3761	1.0876
						0010	0.3743	1.0824
						0100	0.3801	1.0991
						1000	0.3801	1.0991
			0010	0.3458	1.1067	0000	0.4168	1.2052
						0001	0.3743	1.0824
						0010	0.3761	1.0876
						0100	0.3801	1.0991
						1000	0.3801	1.0991
			0100	0.3476	1.1124	0000	0.4209	1.2109
						0001	0.3801	1.0934
						0010	0.3801	1.0934
						0100	0.3815	1.0974
						1000	0.3849	1.1073
			1000	0.3476	1.1124	0000	0.4209	1.2109
						0001	0.3801	1.0934
						0010	0.3801	1.0934
						0100	0.3849	1.1073
						1000	0.3815	1.0974

Table 3.5: Simulation results for the verification of the increasing rate ξ by a simple calculation for the ONEMAX problem for length m, where $m = 4$. t is the time step (or generation), $\mathbf{x}_t$ the observed solution at t, $p_{01}(t)$ the transition probability from s_0 to s_1, and $\xi(t)$ the increasing rate $\frac{p_{01}(t)}{p_{01}(t-1)}$. $p_{01}(t)$ was obtained by the sum of $P[X_t = 1111]$, $P[X_t = 1110]$, $P[X_t = 1101]$, $P[X_t = 1011]$, and $P[X_t = 0111]$.

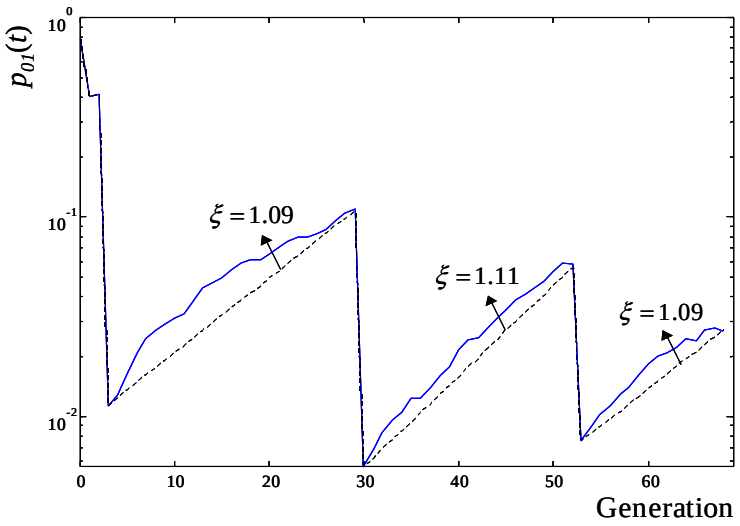
$p_{01}(0) = \frac{5}{2^m} = 0.3125$, since all the solutions have the same probability $\frac{1}{2^m}$ at $t = 0$ and there are five solutions, such as 1111, 1110, 1101, 1011, and 0111, better than 1100. Table 3.5 shows the simulation results of all the possible situations from $t = 1$ to $t = 3$ to verify the value of the increasing rate $\xi(t)$. The rotation angle of p (or $|n|$) for the rotation gate was set to 0.03π in this simple calculation. This table shows that the values of $\xi(t)$ are greater than 1 in all the possible situations. It means that the probability at which the better solution is to be found increases each generation and the better solution can be found in a shorter span of time as shown in (3.12).

Figure 3.19 shows the experimental results of QEA for the ONEMAX problem for length m, where $m = 16$. In Figure 3.19 (a), the dotted line gives a reference for finding a proper ξ which can provide an upper bound of the expected running number of generations for each segment process. If the segment processes of QEA are modelled by SPQEA, the expected running numbers of generations of (3.12) with values of $\xi = 1.09$, 1.1, and 1.09 can provide the upper bound for those of the 2nd, 3rd, and 4th segment processes of QEA, respectively.

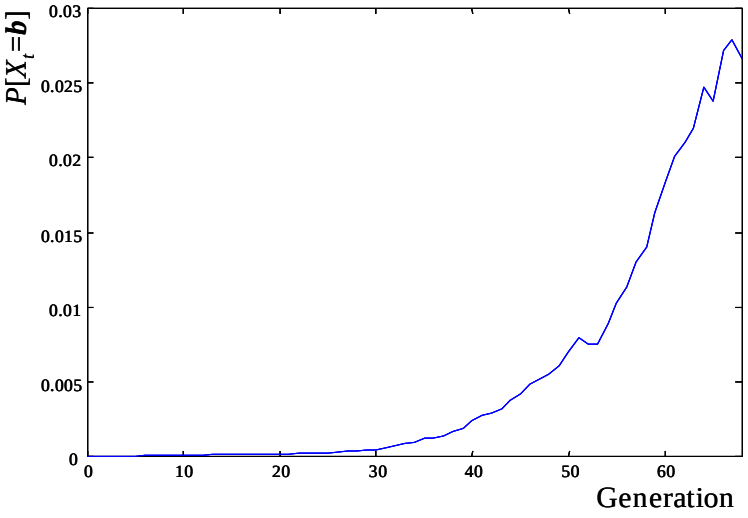
It should be noted that the increasing rate $\xi(t)$ of the transition probability was greater than 1 in the results of Table 3.5 and Figure 3.19. Also, the statement that $\xi(t)$ is always greater than 1 for the ONEMAX problem for length m can be verified by a simple calculation.

Theorem 3.4. *The expected number of Q-bits toward the state 1 for the* ONEMAX *problem is a positive value in SPQEA.*

Proof. Let m be the binary string length and n_1 the number of ones for the current best solution. If the number of ones for the observed binary solution is k, where $k < n_1$, the number of Q-bits toward the state 1 is $(n_1 - k)$ and the number of binary solutions which have k ones is $\frac{m!}{k!(m-k)!}$. Since the number of all the possible binary solutions in SPQEA is $\sum_{k=0}^{n_1-1} \frac{m!}{k!(m-k)!}$, the expected number of Q-bits toward the



(a) Transition probability ($p_{01}(t)$)



(b) Probability of the best solution

Figure 3.19: Experimental results of QEA1 for the ONEMAX problem for length m, where $m = 16$. The dotted line gives a reference for finding a proper ξ which can provide an upper bound of the expected running number of generations for each segment process. A logarithmic (base 10) scale is used for the vertical axis of (a).

state 1 for the ONEMAX problem for length m is a positive value:

$$\frac{\sum_{k=0}^{n_1-1}\left(\frac{m!}{k!(m-k)!}(n_1-k)\right)}{\sum_{k=0}^{n_1-1}\frac{m!}{k!(m-k)!}} > 0. \tag{3.13}$$

In other words, QEA for the ONEMAX problem has the tendency of converging to better solutions in a short span of time. The reason can be explained by the concept of building block which is a small, tightly clustered group of genes [78]. In the case of the ONEMAX problem, the group of ones for the current best solution can be regarded as a building block and the probability of this building block is increased by the rotation gate. As a consequence, the probabilities of the better solutions increase.

It is worthwhile to mention that a sequence of SPQEA for the ONEMAX problem guarantees the global solution in terms of expected running number of generations, since the number of better solutions always decreases after one sequence of SPQEA and it eventually becomes 1 to be considered as the only global solution.

3.6.2 Exploration

To increase the performance of EAs for various optimization problems, exploration as well as exploitation discussed earlier should be considered. The global optimum for a unimodal function which has no local optimum can be exploited without exploration. However, if an EA has no scheme for exploration, the global optimum for a multimodal function which has many local optima is not guaranteed to be found out.

To verify the strategy of exploration for QEA, Shannon entropy [79] can be considered as a measure of the amount of information included in a Q-bit individual.

The entropy of $p(\mathbf{x})$, $\mathbf{x} \in \mathbb{X}$, is described as

$$I(p(\mathbf{x})) = -p(\mathbf{x}) \log_2 p(\mathbf{x}),$$

where $\mathbb{X}$ is a search space, $I(\cdot)$ the entropy (or information) of the probability, and $p(\mathbf{x})$ the probability of $\mathbf{x}$, i.e. $P[X = \mathbf{x}]$. The entropy of the probability distribution for the search space represented by a Q-bit individual is driven to be

$$I(p(\mathbf{x})|\mathbf{x} \in \mathbb{X}) = -\sum_{\mathbf{x} \in \mathbb{X}} p(\mathbf{x}) \log_2 p(\mathbf{x}), \tag{3.14}$$

where

$$p(\mathbf{x}) = \prod_{i=1}^{m} p_i$$

with

$$p_i = \begin{cases} |\alpha_i|^2, & \text{if } x_i = 0 \\ |\beta_i|^2, & \text{if } x_i = 1 \end{cases},$$

where x_i is the ith bit of $\mathbf{x}$ and (α_i, β_i) is the ith Q-bit. It should be noted that the entropy initially has the maximum value of m and it decreases gradually, since each probability of $p(\mathbf{x})$, $\mathbf{x} \in \mathbb{X}$, is shifted with a small amount by the rotation gate as generation advances.

For comparison purpose, let us consider $(1 + 1)$ GA with mutation rate $\frac{1}{m}$, where m is the length of binary solution. Crossover operator cannot be used, since the population size is 1.

Definition 3.6. A *Hamming distance* H of the two binary strings, $\mathbf{x}_1$ and $\mathbf{x}_2$, is

defined as the number of their bitwise-different bits, which is defined as

$$H(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{m} |x_{1i} - x_{2i}|$$

where m is the binary string length.

Theorem 3.5. *The entropy of the probability distribution for the search space represented by* $(1+1)$ *GA is a constant regardless of the generation* t *for* $t > 0$.

Proof. Let $\mathbf{x}$ be the current binary solution, $\mathbf{x}'$ the next binary solution, and h the Hamming distance between $\mathbf{x}$ and $\mathbf{x}'$. If $\mathbf{x}'$ with Hamming distance h from $\mathbf{x}$ is $\mathbf{x}^h$, the probability of $\mathbf{x}^h$ can be described as

$$p(\mathbf{x}^h) = \left(\frac{m-1}{m}\right)^{m-h} \left(\frac{1}{m}\right)^{h},$$

and the number of all the possible $\mathbf{x}^h$ is

$$n(\mathbf{x}^h) = \frac{m!}{h!(m-h)!}.$$

The entropy of the probability distribution for the search space is obtained as

$$\begin{aligned} I(p(\mathbf{x})|\mathbf{x} \in \mathbb{X}) &= -\sum_{\mathbf{x}\in\mathbb{X}} p(\mathbf{x}) \log_2 p(\mathbf{x}) \\ &= -\sum_{h=0}^{m} \left(n(\mathbf{x}^h) p(\mathbf{x}^h) \log_2 p(\mathbf{x}^h) \right). \end{aligned} \tag{3.15}$$

Therefore, the entropy of the probability distribution for the search space represented by $(1+1)$ GA is a constant regardless of the generation t as shown in (3.15).

Let us also consider a simulated annealing (SA) method which is a specific version for binary representation (see Appendix B.1).

Theorem 3.6. *The entropy of the probability distribution for the search space rep-*

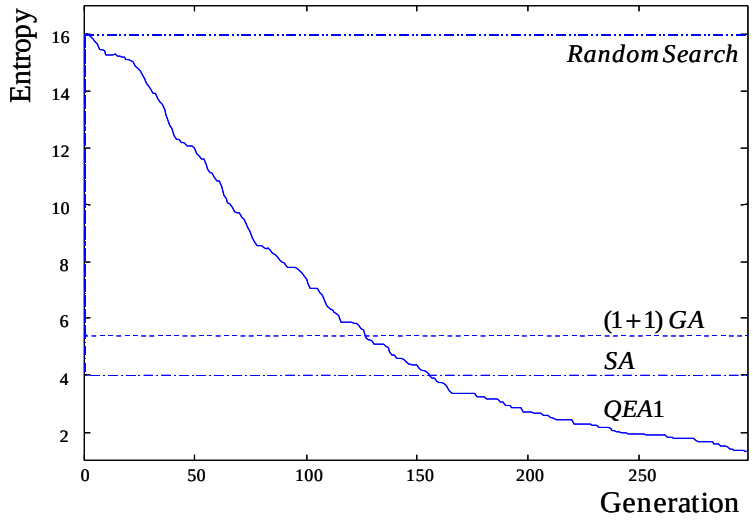


Figure 3.20: Comparison of the entropy of the probability distribution for the search space with respect to the time step (or generation) among QEA1, $(1+1)$ GA, SA, and the random search. The results were obtained from the ONEMAX problem for length m, where $m = 16$.

resented by SA with binary representation is a constant regardless of the time step t for $t > 0$.

Proof. Let $\mathbf{x}$ be the current binary solution, $\mathbf{x}'$ the next solution, and h the Hamming distance between $\mathbf{x}$ and $\mathbf{x}'$. Then the distance h is always 1 for SA with binary representation. If the length of binary string is m, the number of all the possible $\mathbf{x}^1$ is m and the probability of $\mathbf{x}^1$ is $\frac{1}{m}$. Since $p(\mathbf{x}^h)$ is 0 for all h excluding $h = 1$, the entropy of the probability distribution for the search space is obtained as

$$\begin{aligned} I(p(\mathbf{x})|\mathbf{x} \in \mathbb{X}) &= -\sum_{\mathbf{x}\in\mathbb{X}} p(\mathbf{x}) \log_2 p(\mathbf{x}) \\ &= -\log_2 \frac{1}{m}. \end{aligned} \tag{3.16}$$

Therefore, the entropy of the probability distribution for the search space represented by SA with binary representation is a constant regardless of the time step t

as shown in (3.16).

Figure 3.20 shows the differences of the entropy of the probability distribution for the search space among QEA1, $(1+1)$ GA, SA, and the random search. While the entropy values for $(1+1)$ GA, SA, and the random search are constant values of (3.15), (3.16), and m, respectively, that of QEA1 is not a constant. The entropy value of QEA1 is initially the same as that of the random search, and it decreases gradually as generation advances. This result shows clearly that the strategy of QEA for exploration differs from those of $(1+1)$ GA and SA. It is hard to say that which strategy is the superior one compared to others, since the performance of the strategy may depend on the specific problems. However, it is clear that QEA starts with a global search scheme and changes automatically into a local search scheme as generation advances because of its inherent probabilistic mechanism, that leads to a good balance between exploration and exploitation as already mentioned in Section 3.5.

3.7 Termination criteria

To decide the appropriate termination of QEA, a proper termination condition is necessary. Although the maximum number of generations is a generally used termination criterion in evolutionary algorithms, in QEA the probability of the best solution can be employed as a termination criterion because of the probability representation. The termination condition can be designed by using the average probability of the best solution $\mathbf{b}$ as follows:

$$Prob(\mathbf{b}) = \frac{1}{n}\sum_{j=1}^{n}\left(\prod_{i=1}^{m} p_{ji}\right) \tag{3.17}$$

γ_0		0.001	0.01	0.1	0.8	0.9
500	best	3031.2	3036.2	3031.3	3031.3	3036.3
	mean	3008.0	3014.3	3019.0	3018.0	3020.3
	worst	2980.7	2991.1	3006.3	3006.3	3001.3
	σ	12.571	9.798	6.548	6.993	7.895
	t	905	1045	1240	1896	3071

Table 3.6: Experimental results of the knapsack problem with 500 items to show the effects of changing γ_0 for the termination condition (3.18). The population size was 12, the global migration period 100, the local group size 3, and the number of runs 30. σ and t represent the standard deviation and the average number of generations, respectively.

with

$$p_{ji} = \begin{cases} |\alpha_{ji}|^2, & \text{if } b_i = 0 \\ |\beta_{ji}|^2, & \text{if } b_i = 1 \end{cases},$$

where b_i is the ith bit of the best solution $\mathbf{b}$ and $(\alpha_{ji}, \beta_{ji})$ the ith Q-bit of the jth Q-bit individual. The termination condition hence is defined as

$$Prob(\mathbf{b}) > \gamma_0, \tag{3.18}$$

where $0 < \gamma_0 < 1$. The probability $Prob(\mathbf{b})$ represents the average convergence of all the Q-bit individuals to the best solution. It must be a substantial termination criterion of QEA. However, since the probability is sensitive to each Q-bit's probability, it is not easy to set the value γ_0. The slight difference of γ_0 can increase the processing time for a particular problem. Table 3.6 shows the effects of changing the value γ_0. From the table, if $\gamma_0 \geq 0.1$, all the results were almost the same. However, the generation number at $\gamma_0 = 0.9$ was about 2.5 times of that at $\gamma_0 = 0.1$.

To design a new termination criterion regardless of the sensitivity, a measure of

γ		0.9	0.95	0.99
500	best	2979.5	3016.2	3031.3
	mean	2949.5	2993.9	3020.1
	worst	2905.9	2960.7	3001.3
	σ	20.372	14.295	7.681
	t	484	722	1164

Table 3.7: Results on changing γ for the termination criterion (3.19). The parameter settings were the same as in Table 3.6. σ and t represent the standard deviation and the average number of generations, respectively.

the convergence of Q-bit individual is defined.

Definition 3.7. *Q-bit convergence* C_b is defined to be the convergence measure of a Q-bit individual in QEA as

$$C_b(\mathbf{q}) = \frac{1}{m}\sum_{i=1}^{m}\left|1-2|\alpha_i|^2\right|$$

or

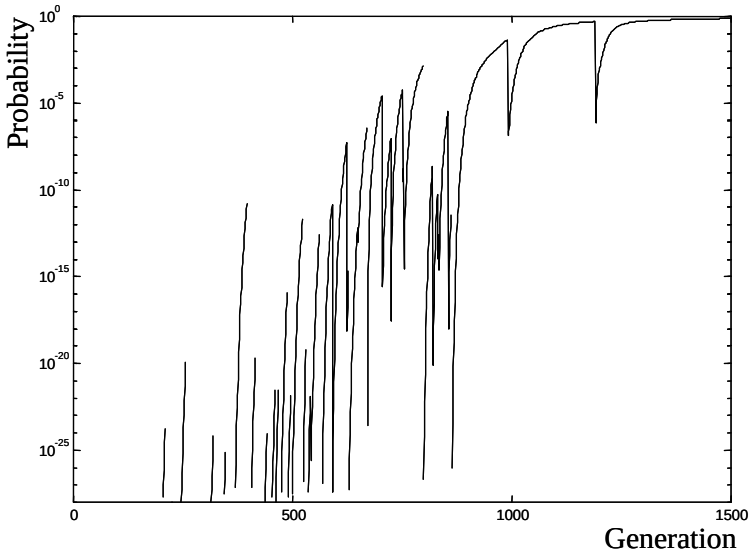
$$C_b(\mathbf{q}) = \frac{1}{m}\sum_{i=1}^{m}\left|1-2|\beta_i|^2\right|,$$

where $\mathbf{q}$ is a Q-bit individual, and (α_i, β_i) is its ith Q-bit.

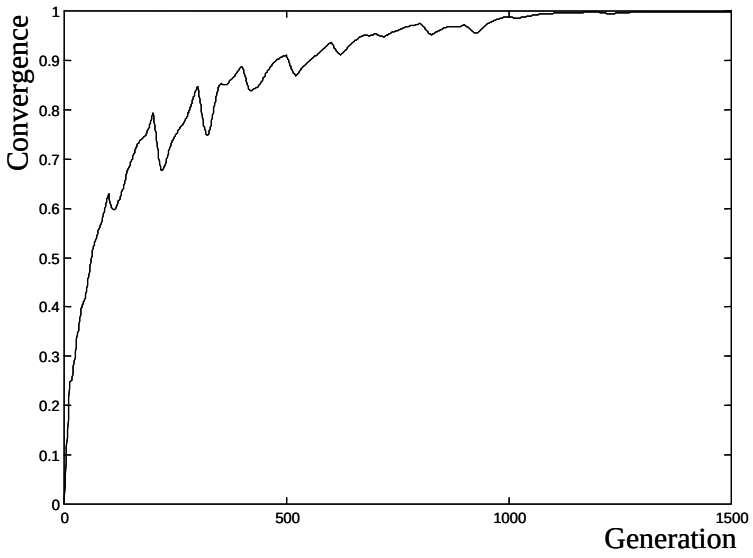
Using the Q-bit convergence, the following termination criterion can be designed:

$$C_{av} = \left(\frac{1}{n}\sum_{j=1}^{n} C_b(\mathbf{q}_j)\right) > \gamma, \tag{3.19}$$

where $C_b(\mathbf{q}_j)$ is the Q-bit convergence of the jth Q-bit individual. The termination criterion of (3.19) is shown to be regardless of the best solution $\mathbf{b}$. However, the average convergence of all the Q-bit individuals can represent the processing status of QEA properly, and it gives a clearer meaning on how much each Q-bit converges to 0 or 1 on an average. Consequently, users can more systematically set the ter-



(a) Mean probability of the best solution: $\log_{10}(Prob(\mathbf{b}))$



(b) Mean Q-bit convergence: C_{av}

Figure 3.21: Difference between the two measures of (3.18) and (3.19) for termination criteria. A logarithmic (base 10) scale is used for the vertical axis of (a).

mination condition. For example, if C_{av} is 0.99, it means that 99% of the Q-bits converge to the true value (0 or 1) on an average. Table 3.7 shows the results on changing the value γ.

Figures 3.21 (a) and (b) show the difference between the two measures of (3.18) and (3.19) for termination criteria. In Figure 3.21 (b), it should be noted that the Q-bit convergence provides an easier understanding of the Q-bit individuals' convergence.

It is worthwhile to mention that if a faster termination is needed, the following maximum Q-bit convergence C_{max} can be used:

$$C_{max} = \left(\max_{j=1}^{n} C_b(\mathbf{q}_j) \right) > \gamma. \tag{3.20}$$

3.8 Effects of different parametric settings

In this section, the effects of changing parameters (such as the population size, the global and local migration periods, the rotation angles, and the number of observations) of QEA are investigated.

3.8.1 Population size

To investigate the effects of changing the population size of QEA, the knapsack problem with 500 items considered in Section 3.3 was used. The population size was tested from 1 to 100. The rotation gate was used for Q-gate. The values of 0.01π, -0.01π, and 0 were used for θ_3, θ_5, and the rest of Θ, respectively. The global migration period in generation was 100, and the local migration period was 1. The local group size n_g was set as

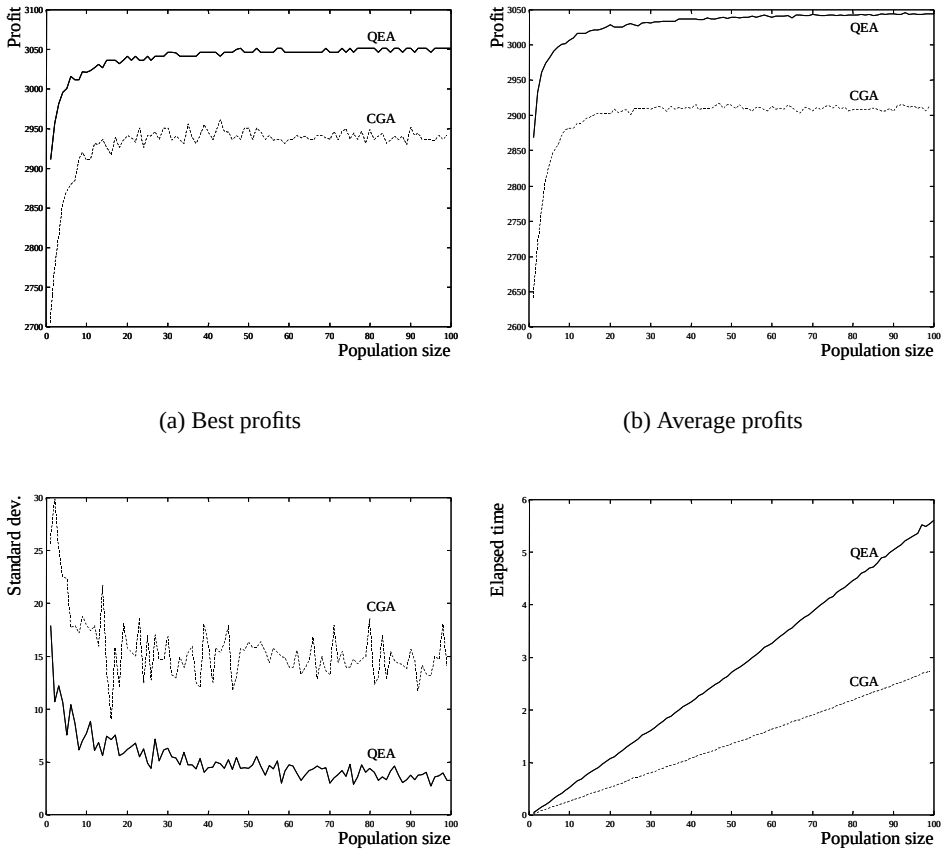
$$n_g = \max\left(integer\left(\frac{n}{5}\right), 1 \right), \tag{3.21}$$

where n is the population size. For the comparison purpose, the conventional GA (CGA) which outperformed all other CGAs in Section 3.3 was tested. The values of 0.001 and 0.7 for the mutation and crossover probabilities, respectively, were selected for CGA ($Rep2$). The maximum number of generations was 1,000.

Figure 3.22 shows the results on the effects of changing the population sizes of QEA and CGA. In Figure 3.22 (a) and (b), the profits increased fast until the population size was 10-20, however the increasing rate was nearly constant after the population size reached 30. The tendency of the results on QEA was similar to that of CGA. However, it should be noted that the best and average profits of QEA with population size 2 were better than those of CGA with population size 100 (according to (d), the convergence speed of QEA with population size 2 was 29 times faster than that of CGA with population size 100). In Figure 3.22 (c), it is also worthwhile to mention that the standard deviation of the best profits of QEA over 30 runs decreased as population size increased. It means that the larger population size could provide better robustness for QEA. However, this relation between population size and robustness did not appear in the result of CGA after the population size reached 20. The processing time of QEA was about 2 times of the CGA's for the same population size. This is because QEA uses Q-bit individuals as a population. Q-bit individuals need floating point calculations to represent the corresponding probabilities. However, it should be noted that the processing time of QEA was proportional to the population size n, which is the same as CGA's.

3.8.2 Global and local migrations

The performance of QEA with a few individuals was already verified in the previous results. It means that QEA can be easily extended to a parallel scheme as proposed in the structure of QEA. Since the parallel scheme can increase the population diversity, it helps QEA to explore the search space effectively.



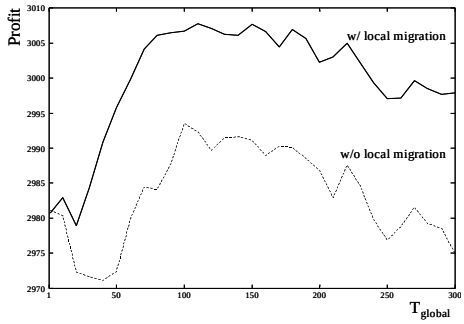
(a) Best profits

(b) Average profits

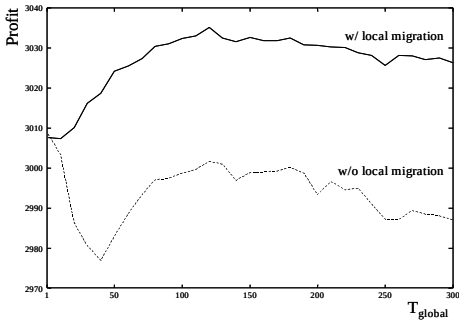
(c) Standard deviation of profits

(d) Processing time (sec .)

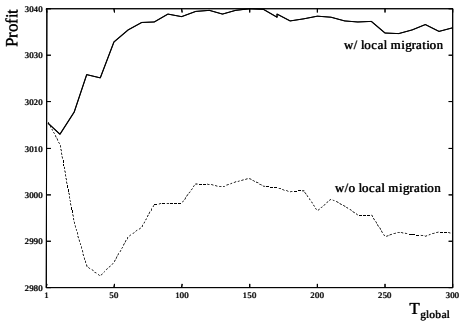
Figure 3.22: Effects of changing the population sizes of QEA and CGA for the knapsack problem with 500 items. The global migration period and the local migration period were 100 and 1, respectively. The local group size was set as (3.21). The results were averaged over 30 runs.



(a) Population size 10

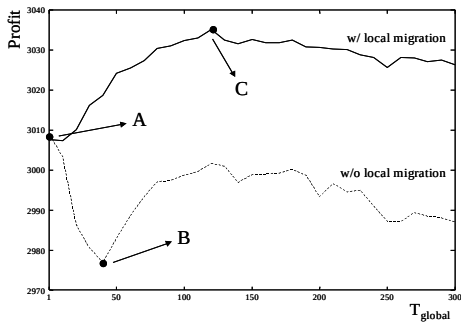


(b) Population size 30

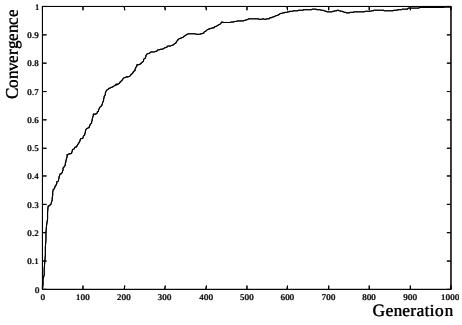


(c) Population size 50

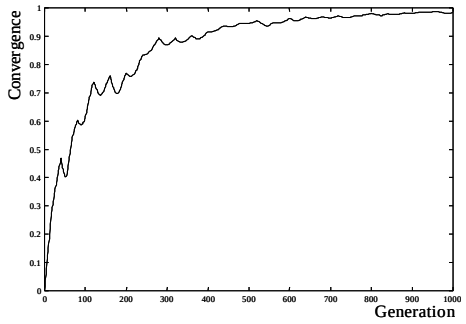
Figure 3.23: Effects of changing the global migration period in QEAs with and without local migration for the knapsack problem with 500 items. The global migration period T_{global} was set to the values ranging from 1 to 300. For the QEA with local migration, the period was 1 and the local group size was set as (3.21). The profits were averaged over 30 runs.



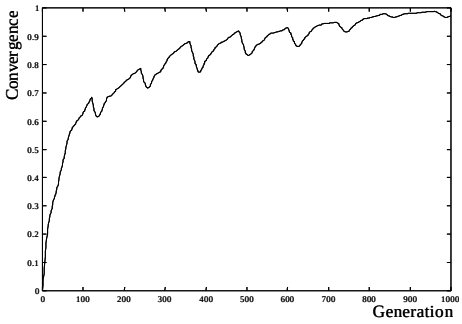
(a) Effects of migrations



(b) Q-bit convergence (A)



(c) Q-bit convergence (B)



(d) Q-bit convergence (C)

Figure 3.24: Relations between migration and Q-bit convergence for the knapsack problem with 500 items. The population size was 30. The global migration periods of (b), (c), and (d) were 1, 40, and 120, respectively. The local group sizes of (b), (c), and (d) were 1, 1, and 6, respectively.

To show the effects of changing the global migration period, the same knapsack problem with 500 items was considered. The population sizes of 10, 30, and 50 were tested. The maximum number of generations was 1,000. To investigate the effects of using the local migration, QEAs with local migration and without local migration were considered for each population size. Figure 3.23 shows the effects of changing the global migration period in QEAs with and without local migration. The local group size of QEA with local migration was set to be the same as (3.21). In the results of QEA without local migration, an undershooting point at near 40 was found, since the increasing diversity from the global migration disturbed the convergence of homogeneous individuals. In the results of QEA with local migration, the undershooting point disappeared. This is because the local migration with period 1 guaranteed the convergence of homogeneous individuals in the same local group. From these results, we can say that it is desirable that the local migration period be set to 1 to guarantee the convergence of homogeneous individuals. It is also worthwhile to mention that the best results were found at the global migration period between 100 and 150, although the migration period could be affected by other parameters. Consequently, the global migration period should be set properly considering the convergence period of the local groups.

Figure 3.24 shows the relations between migration and Q-bit convergence. While the profits of the point A (Figure 3.24 (b)) increased continuously without perturbation, those of the point B (Figure 3.24 (c)) and the point C (Figure 3.24 (d)) increased with perturbation. The perturbation was caused by the global migration. The difference of the perturbation level between the points B and C can be explained by using the concept of Hamming distance. When a new best solution comes from the neighbor local group through the global migration:

i) if the new best solution has a large Hamming distance from the current best solution, the Q-bit individual varies largely to adapt the new one;

ii) if the new best solution has a small Hamming distance from the current best solution, the Q-bit individual changes a little. In this case, the Q-bit individual has a chance to have a premature convergence to a local optimum.

3.8.3 Rotation angles

In the previous empirical results, the best results on the knapsack problem with 500 items were found at the global migration period between 100 and 150. And the rotation angle of p (or $|n|$) was set to 0.01π. However, if the rotation angle is changed, the global migration period for inducing the best result may be changed. If the value of rotation angle is smaller, the global migration period must be larger, since the convergence speed is changed to be slower.

Here, to investigate the effects of changing the rotation angles, the knapsack problems with 500, 600, and 700 items were considered. The population size and the local group size were set to 30 and 6, respectively. The local migration period was set to 1. The termination condition of (3.19) was used instead of MAXGEN and the value of γ was set to 0.99.

Figure 3.25 (a) shows the effects of changing the value of rotation angle ranging from 0.005π to 0.05π. As shown in this figure, it should be noted that there was a peak value of the mean best profits for the same rotation angle δ_θ. And the value of global migration period for the peak was larger as the value of rotation angle was smaller. Figure 3.25 (b) shows the relation between the rotation angle and the global migration period for each peak. And it shows that the rotation angle is inversely proportional to the global migration period. The result was approximately the same as

$$T_g = \frac{1.15\pi}{\delta_\theta} + 10. \tag{3.22}$$

Also, the running number of generations where the algorithm is terminated by the

termination condition for $\gamma = 0.99$ was approximately the same as

$$t_{0.99} = \frac{9.0\pi}{\delta_\theta} + 80. \tag{3.23}$$

Figures 3.26 and 3.27 show the similar results in the relations among the rotation angle, the global migration period, and the running number of generations to those of Figures 3.25.

Consequently, the relation between the rotation angle and the global migration can be approximated as

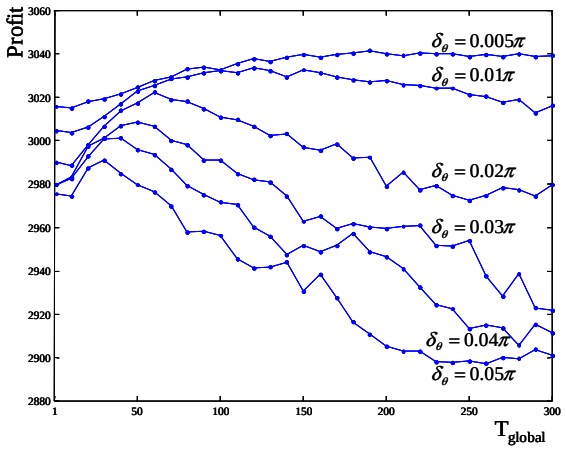
$$T_g = \frac{\lambda_g}{\delta_\theta} + k_g, \tag{3.24}$$

where $\lambda_g > 0$ and $k_g > 0$. The relation between the rotation angle and the running number of generations can be also approximated as

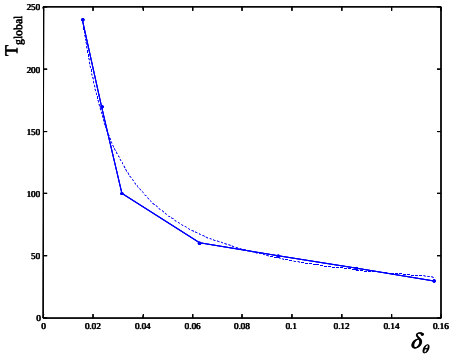
$$t_\gamma = \frac{\lambda_\gamma}{\delta_\theta} + k_\gamma, \tag{3.25}$$

where $\lambda_\gamma > 0$ and $k_\gamma > 0$.

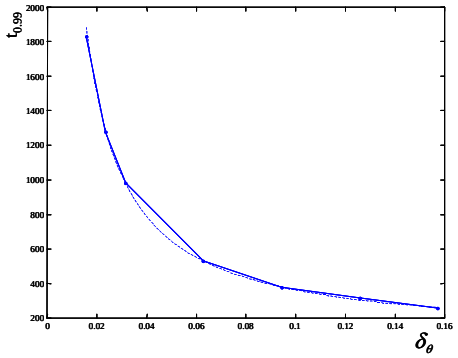
It is worthwhile to mention that k_g of (3.24) and k_γ of (3.25) are nonzero values, since each Q-bit is not updated when the current best solution is changed to the current observed solution in the update procedure. As the result of the knapsack problem with 500 items for checking how many times the current best solution changes during the running number of generations, the best solution $\mathbf{b}$ was changed about 80 times and the best solution $\mathbf{b}_j$ for each individual was changed about 10 times, where the rotation angle and the global migration period were set to 0.01π and 100, respectively.



(a) Mean best profits

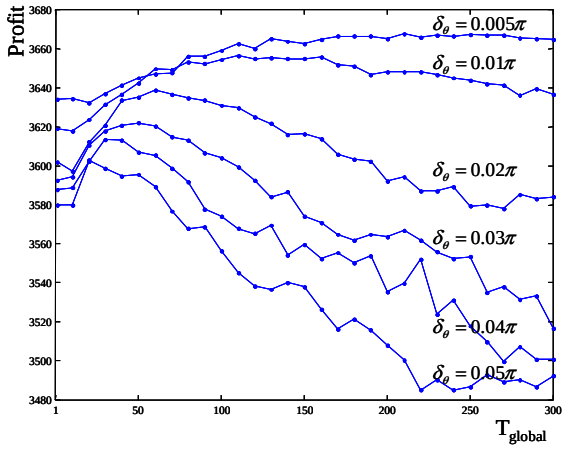


(b) δ_θ and T_g

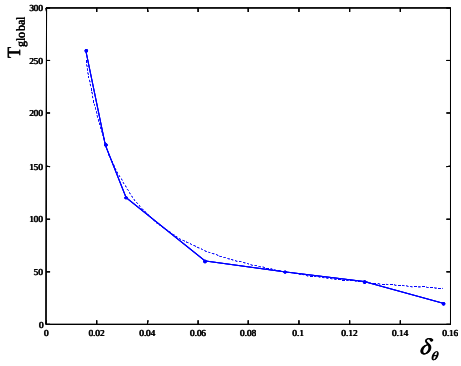


(c) δ_θ and $t_{0.99}$

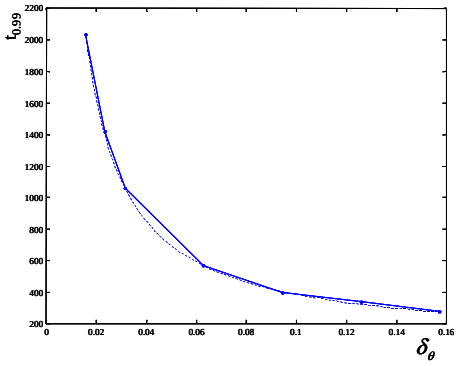
Figure 3.25: Effects of changing the rotation angles for the knapsack problem with 500 items. The global migration period was set to the values ranging from 1 to 300. The population size and the local group size were set to 30 and 6, respectively. The termination condition of (3.19) was used and the value of γ was set to 0.99. All the results were averaged over 30 runs. δ_θ is the rotation angle of p (or $|n|$), T_g the global migration period, and $t_{0.99}$ the number of generations where the algorithm is terminated by the termination condition for $\gamma = 0.99$. The dotted line of (b) is $T_g = \frac{1.15\pi}{\delta_\theta} + 10$ and that of (c) is $t_{0.99} = \frac{9.0\pi}{\delta_\theta} + 80$.



(a) Mean best profits

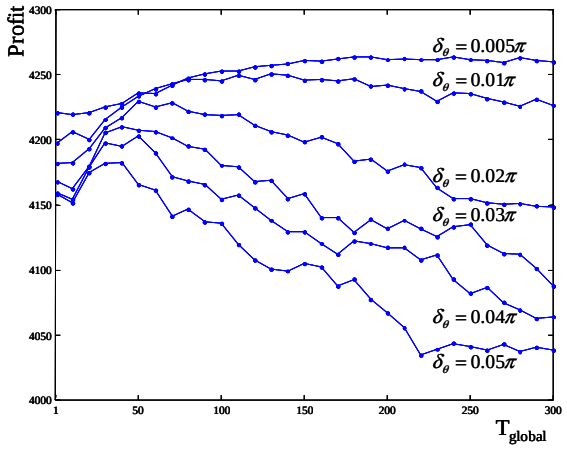


(b) δ_θ and T_g

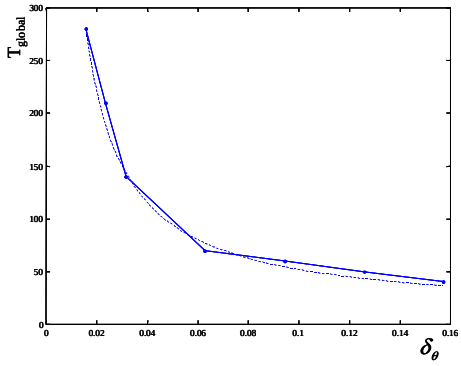


(c) δ_θ and $t_{0.99}$

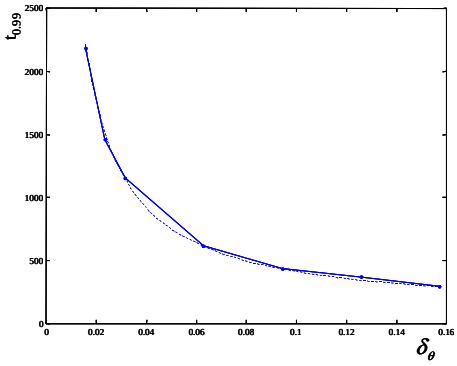
Figure 3.26: Effects of changing the rotation angles for the knapsack problem with 600 items. The global migration period was set to the values ranging from 1 to 300. The population size and the local group size were set to 30 and 6, respectively. The termination condition of (3.19) was used and the value of γ was set to 0.99. All the results were averaged over 30 runs. δ_θ is the rotation angle of p (or $|n|$), T_g the global migration period, and $t_{0.99}$ the number of generations where the algorithm is terminated by the termination condition for $\gamma = 0.99$. The dotted line of (b) is $T_g = \frac{1.2\pi}{\delta_\theta} + 10$ and that of (c) is $t_{0.99} = \frac{9.8\pi}{\delta_\theta} + 80$.



(a) Mean best profits



(b) δ_θ and T_g



(c) δ_θ and $t_{0.99}$

Figure 3.27: Effects of changing the rotation angles for the knapsack problem with 700 items. The global migration period was set to the values ranging from 1 to 300. The population size and the local group size were set to 30 and 6, respectively. The termination condition of (3.19) was used and the value of γ was set to 0.99. All the results were averaged over 30 runs. δ_θ is the rotation angle of p (or $|n|$), T_g the global migration period, and $t_{0.99}$ the number of generations where the algorithm is terminated by the termination condition for $\gamma = 0.99$. The dotted line of (b) is $T_g = \frac{1.34\pi}{\delta_\theta} + 10$ and that of (c) is $t_{0.99} = \frac{10.7\pi}{\delta_\theta} + 80$.

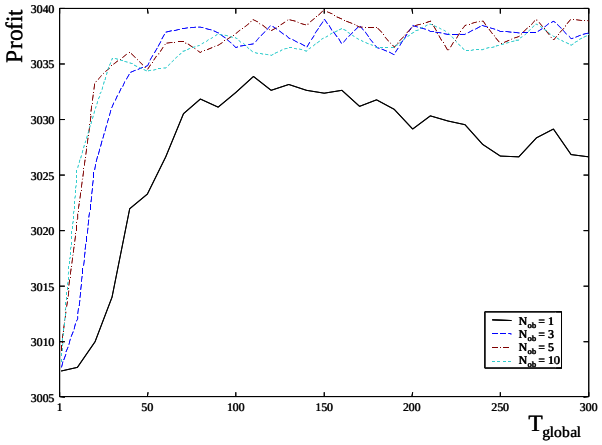


Figure 3.28: Relations between the multiple observations and the global migration period for the knapsack problem with 500 items. The global migration period was set to the values ranging from 1 to 300. The numbers of observations N_{ob} were 1, 3, 5, and 10, respectively. The profits were averaged over 30 runs.

3.8.4 Multiple observations

In QEA, the observation process of a Q-bit individual provides a binary string. Since the Q-bit individual includes many binary strings, instead of the Q-bit individual, the binary string from it is evaluated to give its fitness level. To represent the Q-bit individual, several binary strings can be obtained by multiple observations. In this case, we can guess that multiple observations are related to the convergence of QEA.

To investigate the relations between the number of observations and the performance of QEA, the knapsack problem with 500 items was considered. The population size, the global migration period, and the local group size were set to 10, 100, and 2, respectively.

Figure 3.28 shows the relations between the multiple observations and the global migration period. The global migration period was set to the values ranging from 1 to 300. The numbers of observations were 1, 3, 5, and 10. From this result, it should be noted that the multiple observations increased the performance and decreased the

sensitivity of the global migration period although the profits were almost the same at $N_{ob} = 3$, 5, and 10. However, there is no doubt that the multiple observations increase the processing time. If the complexity of the problem is not high, it is recommended to set the value to 1.

3.9 Summary

In this chapter, a novel QEA, inspired by the concept of quantum computing, was proposed. A Q-bit individual was defined as a string of Q-bits for the probabilistic representation. To introduce the variation to the Q-bit individual, a Q-gate was 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 convergence of the Q-bit individuals. The effectiveness and applicability of QEA were verified by the theoretical analysis of the QEA algorithm as well as the experimental results on several optimization problems.

4. QEA Issues

In this chapter, the structure of QEA is extended for the improvement in its performance. In an attempt to do so, the following issues are addressed: i) the effects of changing the initial values of Q-bits since the initial values can influence the performance of QEA (Note that in the standard QEA, the initial Q-bit is set to $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ for the uniform distribution of states 0 and 1), ii) a novel variation operator H_ϵ gate to provide an attempt to escape effectively from local optima, and iii) a two-phase scheme from the analysis of i).

4.1 Effects of changing the initial values of Q-bits

In the 'initialize $Q(t)$' step of Figure 3.1, all Q-bits are initialized with $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ to represent a linear superposition of all the possible solutions with the same probability. It means that we have no information about the search space. Here, let us assume that we have a little bit of information about the search space to be explored. Then, we can see that the prior knowledge can be easily put into the initial values of Q-bits.

For instance, let us consider the knapsack problem with 500 items, which does not have an average knapsack capacity as a constraint, but a restrictive knapsack capacity of $C = 2v$, where $v = 10$. In this case, the optimal solution contains very few items. An infeasible search space, where the constraint is not satisfied, occupies almost the whole search space. For this type of knapsack problem, we already have some prior knowledge like "the optimal solution contains very few items." From

(α_i^2, β_i^2)		(0.99, 0.01)	(0.5, 0.5)	(0.01, 0.99)
500	b.	94.993	94.856	94.892
	m.	86.973	81.978	80.969
	w.	79.942	69.983	69.996
	σ	3.774	5.846	5.662
	t	697	4682	8231

Table 4.1: Experimental results of the knapsack problem with 500 items to show the effects of changing the initial values of Q-bits. The population size was 15, the global migration period 100, the local group size 3, and the number of runs 30. The termination condition of (3.19) was used with $\gamma = 0.99$. $b.$, $m.$, and $w.$ mean $best$, $mean$, and $worst$, respectively. σ and t represent the standard deviation and the average number of generations, respectively.

this prior knowledge, we can set the initial value of each Q-bit to $\left(\sqrt{1-\beta_i^2}, \beta_i\right)$, where β_i is a small value close to zero.

Table 4.1 shows the experimental results of the knapsack problem with 500 items using the knapsack capacity to be restricted to $C = 20$. The table shows that the results are highly dependent on the initial values of Q-bits. The results of $(0.99, 0.01)$ are the best and also its average number of generations is the smallest one. More specifically, the convergence speed of $(0.99, 0.01)$ is 6.7 and 11.8 times faster than those of $(0.5, 0.5)$ and $(0.01, 0.99)$, respectively. In addition to that, the average standard deviation of $(0.99, 0.01)$ over 30 runs is the best one.

The results of Table 4.1 agree with our prediction through the prior knowledge. It can be explained by the relation between the initial search space and the optimal solution. If the initial search space is formed near the optimal solution, the solution can be searched in a short span of time. To measure the distance between the initial search space and the optimal solution, ones-number distance, defined below, can be considered.

Definition 4.1. A *ones-number distance*, D_n, of the two binary strings, $\mathbf{x}_1$ and $\mathbf{x}_2$,

is defined as the difference between their numbers of ones, which is defined as

$$D_n(\mathbf{x}_1, \mathbf{x}_2) = |ones(\mathbf{x}_1) - ones(\mathbf{x}_2)|,$$

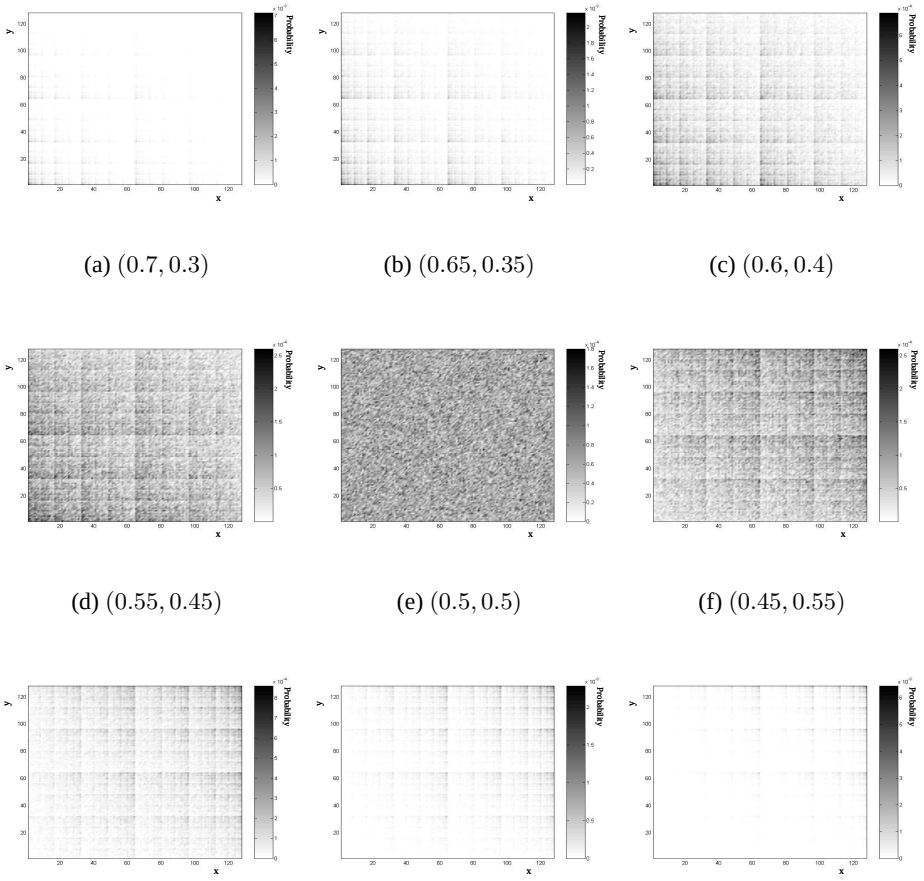
where the function $ones(\mathbf{x})$ returns the number of ones in the binary string $\mathbf{x}$.

The distribution of the ones-number distance between the initial search space and the optimal solution can be obtained, since the initial search space determined by each value of Q-bit has a distribution with respect to ones number.

Figure 4.1 shows the differences of the initial search spaces with respect to the initial values of Q-bits. In the case of (e), the distribution of the initial search space is nearly a random noise. It indicates that the initial search starts randomly. In the cases of (a)-(d), the points which include less ones have higher probabilities. On the other hand, in the cases of (f)-(i), the points which include more ones have higher probabilities. It means that the initial search space can be formed effectively by changing the initial values of Q-bits. It is worthwhile to mention that the initial search space is distributed globally, although the distribution spreads to the space including more (or less) ones depending on Q-bit values. For instance, let us consider 7-bit strings with 1 for the number of ones. There are 7 strings of which number of ones is 1. Their integer numbers are 1, 2, 4, 8, 16, 32, and 64, respectively. It indicates that the solutions which have the same number of ones spread widely in the search space. The reason why the initial search space represented by the initial values of Q-bits is distributed globally can be explained by this characteristics of binary coding.

Figure 4.2 shows the differences of the initial search spaces with respect to the initial values of Q-bits, when gray coding is used to convert binary string to integer value.

Figure 4.3 shows the observed frequency of each solution with respect to the number of ones by observing each Q-bit individual 10^5 times. The results show

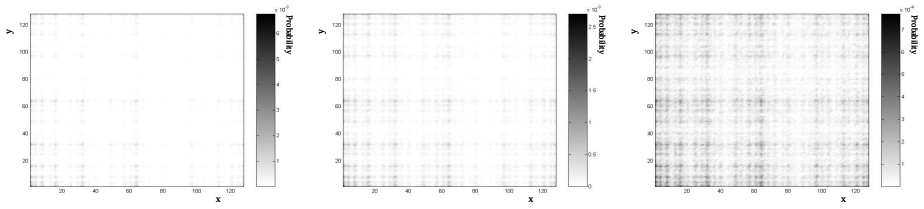


(a) $(0.7, 0.3)$ (b) $(0.65, 0.35)$ (c) $(0.6, 0.4)$

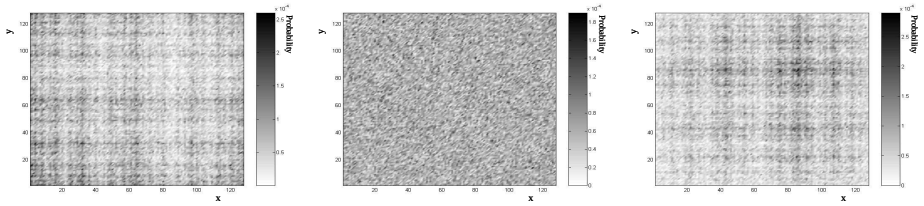
(d) $(0.55, 0.45)$ (e) $(0.5, 0.5)$ (f) $(0.45, 0.55)$

(g) $(0.4, 0.6)$ (h) $(0.35, 0.65)$ (i) $(0.3, 0.7)$

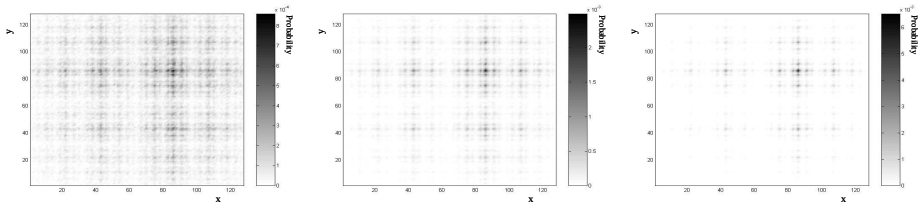
Figure 4.1: Differences in the initial search spaces with respect to the initial values of Q-bits. The size of the search space is 2^{14} (7 bits for each x and y). Each pair of values in parenthesis represents (α_i^2, β_i^2). On the (x, y) plane, the darker points have a higher probability to be present in the initial search space. The probabilities were obtained by observing each Q-bit individual 10^5 times.



(a) $(0.7, 0.3)$ (b) $(0.65, 0.35)$ (c) $(0.6, 0.4)$

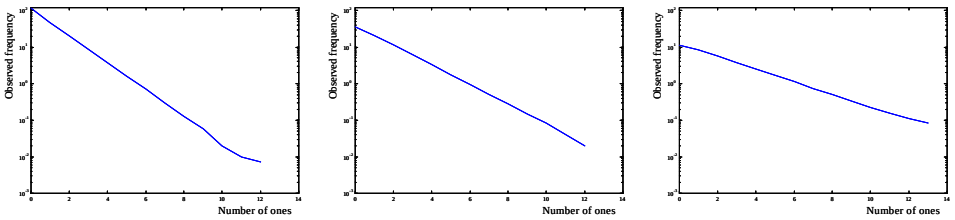


(d) $(0.55, 0.45)$ (e) $(0.5, 0.5)$ (f) $(0.45, 0.55)$

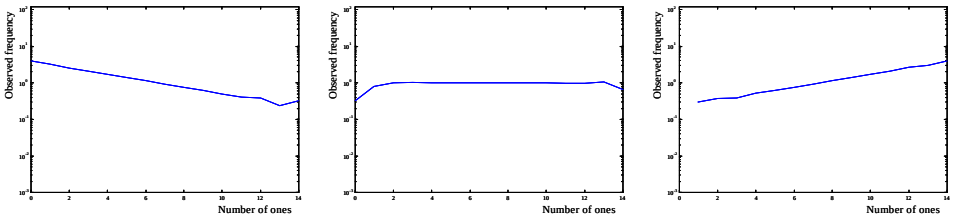


(g) $(0.4, 0.6)$ (h) $(0.35, 0.65)$ (i) $(0.3, 0.7)$

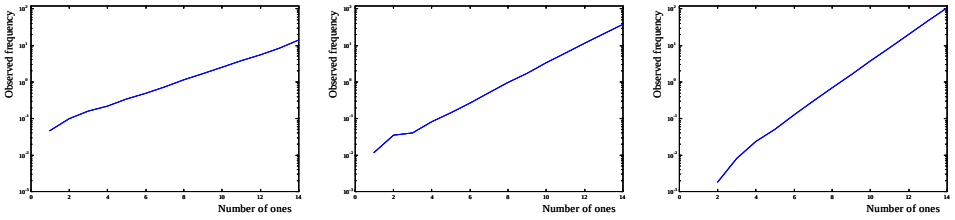
Figure 4.2: Differences in the initial search spaces with respect to the initial values of Q-bits. The size of the search space is 2^{14} (7 bits for each x and y). Each pair of values in parenthesis represents (α_i^2, β_i^2). On the (x, y) plane, the darker points have a higher probability to be present in the initial search space. The probabilities were obtained by observing each Q-bit individual 10^5 times. In particular, gray coding was used to convert from binary string to integer value.



(a) $(0.7, 0.3)$ (b) $(0.65, 0.35)$ (c) $(0.6, 0.4)$



(d) $(0.55, 0.45)$ (e) $(0.5, 0.5)$ (f) $(0.45, 0.55)$



(g) $(0.4, 0.6)$ (h) $(0.35, 0.65)$ (i) $(0.3, 0.7)$

Figure 4.3: Observed frequency of each solution with respect to the number of ones by observing each Q-bit individual 10^5 times. Each pair of values in parenthesis represents (α_i^2, β_i^2). The size of the search space is 2^{14} (7 bits for each x and y). A logarithmic (base 10) scale is used for the vertical axis.

clearly that the initial search space can be formed effectively by changing the initial values of Q-bits. The frequencies were scaled by using the following equation:

$$f_s(n_1) = \frac{f_o(n_1)/N_o}{\frac{m!}{n_1!(m-n_1)!}/2^m} = \frac{f_o(n_1)/10^5}{\frac{14!}{n_1!(14-n_1)!}/2^{14}}, \tag{4.1}$$

where n_1 is the number of ones, f_s the scaled frequency, f_o the observed frequency, N_o the total number of observations, and m the length of binary solution. The value of $\frac{m!}{n_1!(m-n_1)!}$ represents the number of solutions including n_1 ones among the whole solutions.

4.2 H_ϵ gate

The rotation gate used as a Q-gate induces the convergence of each Q-bit to either 0 or 1. However, a Q-bit converged to either 0 or 1 cannot escape the state by itself, although it can be changed passively by a global or local migration. If the value of $|\beta|^2$ is 0 (or 1), the observing state of the Q-bit is always 0 (or 1). To prevent the premature convergence of Q-bit, H_ϵ gate is defined as a Q-gate.

Definition 4.2. An *H_ϵ gate* is defined as a Q-gate extended from the rotation gate:

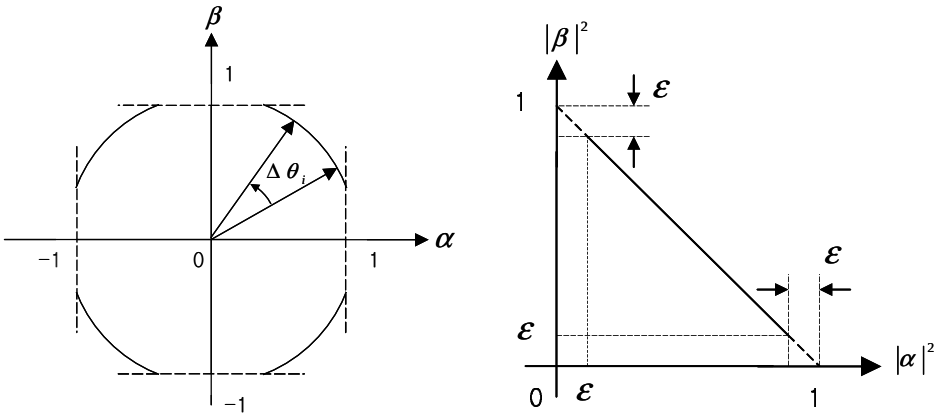
$$[\alpha_i' \ \beta_i']^T = H_\epsilon(\alpha_i, \ \beta_i, \ \Delta\theta_i), \tag{4.2}$$

where, for $[\alpha_i'' \ \beta_i'']^T = R(\Delta\theta_i)[\alpha_i \ \beta_i]^T$,
i) if $|\alpha_i''|^2 \leq \epsilon$ and $|\beta_i''|^2 \geq 1-\epsilon$,

$$[\alpha_i' \ \beta_i']^T = [\sqrt{\epsilon} \ \sqrt{1-\epsilon}]^T;$$

ii) if $|\alpha_i''|^2 \geq 1-\epsilon$ and $|\beta_i''|^2 \leq \epsilon$,

$$[\alpha_i' \ \beta_i']^T = [\sqrt{1-\epsilon} \ \sqrt{\epsilon}]^T;$$



(a) H_ϵ gate (b) ϵ constraints

Figure 4.4: H_ϵ gate based on the rotation gate.

iii) otherwise,

$$[\alpha_i' \ \beta_i']^T = [\alpha_i'' \ \beta_i'']^T,$$

where $0 < \epsilon \ll 1$, $R(\Delta\theta_i)$ is the rotation gate, and $\Delta\theta_i$, $i = 1, 2, \cdots, m$, is the rotation angle of each Q-bit toward either 0 or 1 state, depending on its sign.

Figure 4.4 shows the H_ϵ gate, where $\lim_{\epsilon \to 0} H_\epsilon(\cdot)$ is the same as the rotation gate. While the rotation gate makes the probability of $|\alpha|^2$ or $|\beta|^2$ converge to either 0 or 1, H_ϵ gate makes it converge to ϵ or $(1 - \epsilon)$. It should be noted that if ϵ is too big, the convergence tendency of a Q-bit individual may disappear.

Theorem 4.1. *The entropy of the probability distribution for the search space represented by Q-bit individual with H_ϵ gate converges to*

$$-\sum_{h=0}^{m} \left(\frac{m!}{h!(m-h)!} \epsilon^{m-h} (1-\epsilon)^h \left(\log_2 \left(\epsilon^{m-h} (1-\epsilon)^h \right) \right) \right), \tag{4.3}$$

where m is the length of Q-bit individual.

Proof. Let $\mathbf{x}$ be the converged binary solution and $\mathbf{x}^h$ the binary solution with Hamming distance h from $\mathbf{x}$. Each Q-bit converges to either $(\sqrt{\epsilon}, \sqrt{1-\epsilon})$ or $(\sqrt{1-\epsilon}, \sqrt{\epsilon})$ in any case. For clarification purpose, let us consider one simple case, where all Q-bits corresponding to $\mathbf{x}$ converge to $(\alpha, \beta) = (\sqrt{\epsilon}, \sqrt{1-\epsilon})$. Then the probability of $\mathbf{x}^h$ is described as

$$p(\mathbf{x}^h) = \epsilon^{m-h}(1-\epsilon)^h,$$

and the number of all the possible $\mathbf{x}^h$ is

$$n(\mathbf{x}^h) = \frac{m!}{h!(m-h)!}.$$

Therefore, the entropy of the probability distribution for the search space represented by the Q-bit individual with H_ϵ gate is obtained as

$$\begin{aligned} I(p(\mathbf{x})|\mathbf{x} \in \mathbb{X}) &= -\sum_{\mathbf{x}\in\mathbb{X}} p(\mathbf{x}) \log_2 p(\mathbf{x}) \\ &= -\sum_{h=0}^{m} \Big(n(\mathbf{x}^h) p(\mathbf{x}^h) \log_2 p(\mathbf{x}^h) \Big) \\ &= -\sum_{h=0}^{m} \left(\frac{m!}{h!(m-h)!} \epsilon^{m-h}(1-\epsilon)^h \left(\log_2 \left(\epsilon^{m-h}(1-\epsilon)^h \right) \right) \right). \end{aligned}$$

Figure 4.5 shows the differences of the entropy of the probability distribution for the search space among QEA1s with H_ϵ gates for $\epsilon = 0$, 0.01, and 0.05, respectively. From the results for the ONEMAX problem of (3.9), it should be noted that the entropy converges to the larger value as ϵ is bigger. However, if ϵ is too big, the mechanism for exploitation may not work.

If H_ϵ gate is used as a Q-gate, the termination conditions of (3.19) and (3.20)

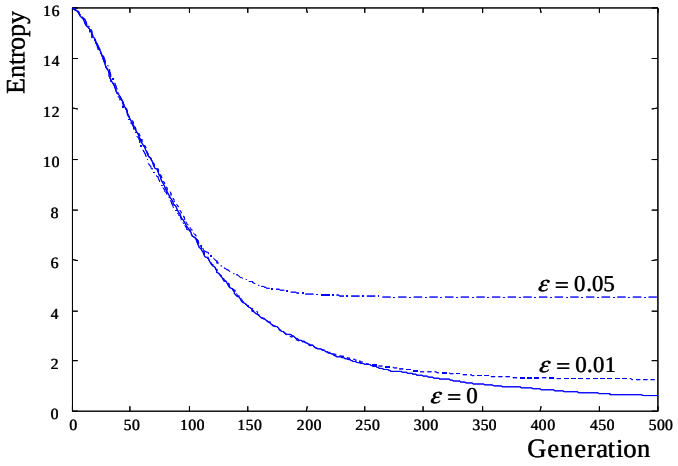


Figure 4.5: Comparison of the entropy of the probability distribution for the search space among QEA1s with H_ϵ gates for $\epsilon = 0$, 0.01, and 0.05, respectively. All the results were averaged over 30 runs for the ONEMAX problem for length m, where $m = 16$.

should be modified as

$$C_{av} = \left(\frac{1}{n} \sum_{j=1}^{n} C_b(\mathbf{q}_j) \right) > (1 - 2\epsilon)\gamma \tag{4.4}$$

and

$$C_{max} = \left(\max_{j=1}^{n} C_b(\mathbf{q}_j) \right) > (1 - 2\epsilon)\gamma, \tag{4.5}$$

respectively. To increase the period for fine tuning caused by the ϵ boundary, the mixed termination condition can be also used as follows:

$$\text{MAXGEN} = \tau t_\gamma, \tag{4.6}$$

where t_γ is the number of generations when the termination condition with γ of (4.4) or (4.5) is satisfied and $\tau > 1$.

ϵ		0	0.005	0.01	0.015	0.02	0.03
f	b.	1677.9	3.8×10^{-4}	3.8×10^{-4}	2.6×10^{-3}	0.2841	500.90
	m.	2467.5	31.583	4.2×10^{-4}	7.9145	67.980	1046.6
	w.	3624.3	236.87	7.5×10^{-4}	118.45	475.05	1763.1
	σ	503.83	60.649	7.2×10^{-5}	29.541	104.58	343.73
	t	4567.5	7183.9	8817.2	7364.0	6207.0	2903.2

Table 4.2: Experimental results of the Schwefel function to show the effects of changing ϵ for H_ϵ gate. The population size was 15, the global migration period 100, the local group size 3, the number of observations 3, and the number of runs 30. γ of (4.4) was set to 0.9999. $b.$, $m.$, and $w.$ mean $best$, $mean$, and $worst$, respectively. σ and t represent the standard deviation and the average number of generations, respectively.

To investigate the performance of H_ϵ gate, Schwefel function (see Appendix A.2) is considered. Table 4.2 shows the effects of changing ϵ for the H_ϵ gate. As shown in the table, the results for $\epsilon = 0.01$ were the best for the Schwefel function, although the average number of generations was larger than other results. It should be noted that if ϵ is too big, the performance would be worse than that of QEA with the rotation gate ($\epsilon = 0$). While a large ϵ ($= 0.03$) induces a fast premature convergence, a properly selected-small value of ϵ ($= 0.01$) provides better solutions. In particular, H_ϵ gate is recommended for a class of numerical optimization problems which have many local optima.

4.3 Two-phase scheme

We have already verified that changing the initial values of Q-bits can provide better performance of QEA. The initial values of Q-bits are directly connected to the initial search space as shown in Figure 4.1. If the initial values of Q-bits can be found to represent the initial search space with small distance to the best solution, the Q-bit individuals can converge to the best solution effectively. To put this idea to the algorithm, two-phase QEA (TPQEA) scheme is proposed in the following.

Procedure TPQEA
begin
First-phase QEA
Second-phase QEA
end

Figure 4.6: Procedure TPQEA.

Procedure first-phase QEA (phase I)
begin
$t \leftarrow 0$
initialize $Q(t)$
make $P(t)$ by observing the states of $Q(t)$
evaluate $P(t)$
store the best solutions among $P(t)$ into $B(t)$
while (**not** termination condition) **do**
begin
$t \leftarrow t + 1$
make $P(t)$ by observing the states of $Q(t-1)$
evaluate $P(t)$
update $Q(t)$ using Q-gates
store the best solutions among $B(t-1)$ and $P(t)$ into $B(t)$
if (local migration condition)
then migrate $\mathbf{b}_j^t$ to $B(t)$ locally
end
store the initial value of Q-bit inducing the best result into $[\alpha^b \ \beta^b]^T$
end

Figure 4.7: First-phase QEA.

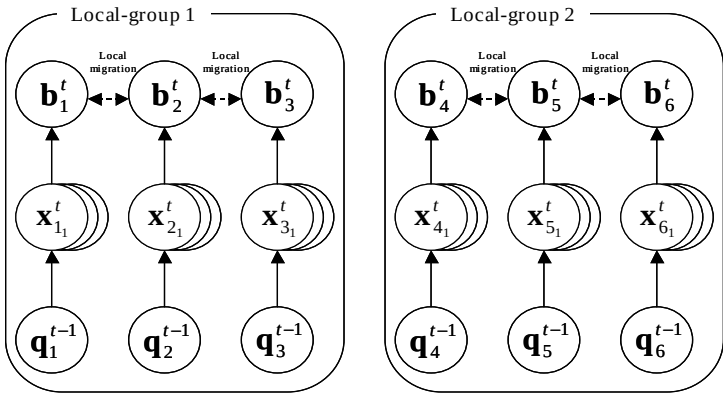


Figure 4.8: Relations between variables for the first-phase QEA, when the population size, the local group size, and the number of observations are 6, 3, and 3, respectively.

TPQEA has two procedures as shown in Figure 4.6. In the first phase as shown in Figure 4.7, a promising initial value is searched and stored into $[\alpha^b \ \beta^b]^T$. The first phase is similar to the standard procedure of QEA except the followings:

i) The 'initialize $Q(t)$' step is different. In this step, each local group has a different value of Q-bit from other local groups to explore a different search space each. In the gth local group, the initial value of Q-bit can be assigned as

$$\begin{bmatrix} \alpha_g \\ \beta_g \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{(1-2\delta)}{N_g-1}g+\delta} \\ \sqrt{1-\frac{(1-2\delta)}{N_g-1}g+\delta} \end{bmatrix}, \tag{4.7}$$

where N_g is the number of local groups, and δ, $0 < \delta \ll 1$, is the minimum probability of the state 1 (or 0). Equation (4.7) assigns a probability of each group dividing an interval $[\delta, 1-\delta]$ into N_g equal parts.

To guarantee the homogeneity of each group, the best solution $\mathbf{b}$ is not used and the global migration process is removed from the standard structure of QEA. For example, the relations between variables are shown in Figure 4.8 when the population size, the local group size, and the number of observations are 6, 3, and 3,

Procedure second-phase QEA (phase II)
begin
initialize $Q(t)$ using $[\alpha^b\ \beta^b]^T$
make $P(t)$ by observing the states of $Q(t)$
evaluate $P(t)$
store the best solutions among $P(t)$ into $B(t)$
while (**not** termination-condition) **do**
begin
$t \leftarrow t + 1$
make $P(t)$ by observing the states of $Q(t-1)$
evaluate $P(t)$
update $Q(t)$ using Q-gates
store the best solutions among $B(t-1)$ and $P(t)$ into $B(t)$
store the best solution $\mathbf{b}$ among $B(t)$
if (global migration condition)
then migrate $\mathbf{b}$ to $B(t)$ globally
else if (local migration condition)
then migrate $\mathbf{b}_j^t$ in $B(t)$ to $B(t)$ locally
end
end

Figure 4.9: Second-phase QEA.

respectively.

ii) The condition of (3.19) (or (4.4)) can be used as a termination condition for the first phase. However, if faster transition from the first phase to the second phase is required, the termination condition of (3.20) (or (4.5)) can be used for the first phase.

iii) At the end of the first phase, a process is added to store the initial value of Q-bit inducing the best result into $[\alpha^b\ \beta^b]^T$.

Figure 4.9 depicts the second phase of TPQEA. It is almost the same as the procedure of QEA, except that time initialization '$t \leftarrow 0$' is removed and the 'initialize $Q(t)$' step shall use the initial value of Q-bit, i.e. $[\alpha^b\ \beta^b]^T$, obtained from the first

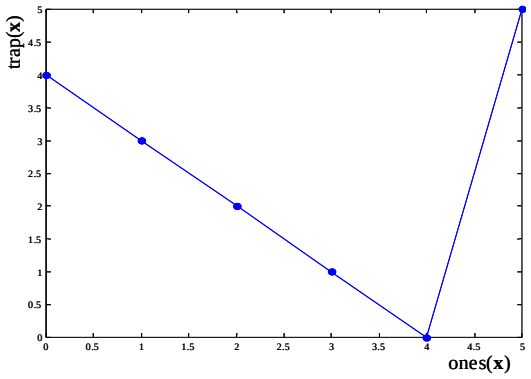


Figure 4.10: 5-bit trap.

phase.

Let us consider the concatenated 5-bit traps as

$$f_{trap}(\mathbf{x}) = \sum_{i=0}^{N_{trap}-1} trap(x_{5i+1}, x_{5i+2}, x_{5i+3}, x_{5i+4}, x_{5i+5}),$$

where N_{trap} is the number of traps and

$$trap(\mathbf{x}) = \begin{cases} 4 - ones(\mathbf{x}) & , \text{ if } ones(\mathbf{x}) \leq 4 \\ 5 & , \text{ if } ones(\mathbf{x}) = 5. \end{cases}$$

To maximize f_{trap}, the individuals should be able to escape from the 5-bit traps, at $(0, 0, 0, 0, 0)$, as shown in Figure 4.10. Table 4.3 shows the results of QEA and TPQEA for the concatenated 5-bit traps with $N_{trap} = 20$. The global maximum value of f_{trap} is 100 when all the 100 bits are ones. While QEA fell into the trap point $(0, 0, 0, 0, 0)$ in 15 traps on an average, TPQEA did not fall into the traps at all. In particular, the average number of generations of TPQEA was smaller than QEA's, although TPQEA has two phases. However, it should be noted that TPQEA may need a larger number of generations to find the best solution for a particular problem as compared to QEA's, if the best solution is included in the search space

		QEA	TPQEA
f_{trap}	best	90	100
	mean	85.033	100
	worst	81	100
	σ	1.722	0
	t	218	84

Table 4.3: Experimental results of the concatenated 5-bit traps with $N_{trap} = 20$. The population size was 15, the global migration period 100, the local group size 3, and the number of runs 30. The termination condition of (4.4) was used with $\gamma = 0.99$. γ and δ for the first phase of TPQEA were 0.9 and 0.05, respectively. ϵ of the H_ϵ gate was 0.01. σ and t represent the standard deviation and the average number of generations, respectively.

with ones-number distance 0 from the initial search space defined with the initial value of Q-bit, $[\frac{1}{\sqrt{2}}\ \frac{1}{\sqrt{2}}]^T$.

4.4 Summary

In this chapter, the structure as well as some basic issues of QEA were studied to improve its performance. In the basic QEA, the initial Q-bit was set to $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$. However, since the initial values could influence the performance of QEA, the effects of changing the initial values of Q-bits were investigated. A modified Q-gate of the rotation gate, H_ϵ gate which is suitable for a class of numerical optimization problems with many local optima, was proposed that provides a scheme to escape from local optima. And a two-phase QEA (TPQEA) scheme was also proposed for a class of optimization problems with the global optimum which is present in the search space with larger ones-number distance from randomly generated search space. In particular, the experimental results of Schwefel function and the concatenated 5-bit traps showed the effectiveness and applicability of the H_ϵ gate and TPQEA, respectively.

5. Experiments

In this chapter, several numerical and combinatorial optimization problems are discussed to demonstrate the effectiveness and applicability of QEA.

5.1 Numerical optimization

5.1.1 Rotation and H_ϵ gates

The representation of real number may be more suitable for numerical optimization than that of binary string. Nevertheless, the six numerical optimization functions of Sphere, Ackley, Griewank, Rastrigin, Schwefel, and Rosenbrock (see Appendix A.2) were considered to demonstrate the effectiveness of QEA to a class of numerical optimization.

To minimize the six functions, QEAs were tested using the parameter settings as given in Table 5.1. And the global migration was not used. Considering the resolutions of variables, the numbers of the Q-bits for the six functions were set to 18, 18, 21, 17, 22, and 18 bits (per variable), respectively. Gray coding was used to convert from binary string to real value. For the comparison purpose, the termination condition with the maximum number of generations was used, since the experiments referred from [80] were tested with a fixed number of generations. The rotation angles of Θ were set to $[0\,0\,p\,0\,n\,0\,0\,0]^T$ as in Section 3.4, where p and $|n|$ (absolute value of n) were set to 0.06π, 0.06π, 0.06π, 0.04π, 0.04π, and 0.04π for the six functions, respectively.

The results taken from the reference [80] were tested using classical EP (CEP)

	QEA with H_ϵ gate	QEA with Rotation gate
Population size	100	100
Number of observations	1	1
Local group size	100	100
ϵ for H_ϵ gate	0.01	(0)
Termination condition	MAXGEN	MAXGEN

Table 5.1: Parameter settings of QEAs for the experiments on the numerical optimization functions (A.1)-(A.6).

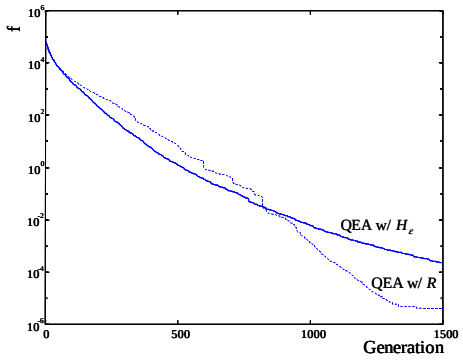
and fast EP (FEP). The population size was selected to be 100 for each experiment. According to the reference, FEP provided better solutions than CEP.

Table 5.2 shows the results of the numerical functions (A.1)-(A.6) for QEA with the H_ϵ gate, QEA with the rotation gate, FEP, and CEP. In the cases of f_{Sphere} and f_{Ackley} which are relatively simple functions compared to the other functions, QEAs and EP had almost the same results, although the results of QEA with the rotation gate were slightly better than the others'. However, in the cases of $f_{Griewank}$ and $f_{Rastrigin}$ which have many local optima, QEA with the H_ϵ gate and FEP had better results than the others. In particular, in the case of $f_{Schwefel}$, only QEA with the H_ϵ gate found the global solution. In the case of $f_{Rosenbrock}$, no algorithm found the global solution on an average. The reason why QEAs could not find the global solution of $f_{Rosenbrock}$ is described in Appendix A.3.

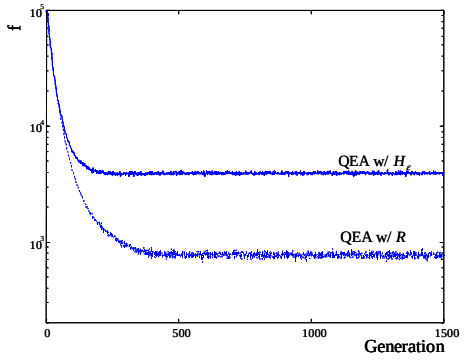
Figures 5.1-5.5 show the comparison between QEA with the H_ϵ gate and QEA with the rotation gate on Sphere, Ackley, Griewank, Rastrigin, and Schwefel functions. In the results of Figures 5.1 and 5.2, the best results of QEA with the rotation gate were better compared to those of QEA with the H_ϵ gate after the generation reached 800. In the results of Figures 5.3, 5.4, and 5.5, however, it should be noted that the best results of QEA with the H_ϵ gate were significantly better compared to those of QEA with the rotation gate, although the average results of QEA with the

		QEAs		EP	
		QEA w/ H_ϵ (100)	QEA w/ R (100)	FEP (100)	CEP (100)
f_{Sphere}	m.	1.8×10^{-4}	4.3×10^{-6}	5.7×10^{-4}	2.2×10^{-4}
$t = 1500$	σ	1.3×10^{-4}	0	1.3×10^{-4}	5.9×10^{-4}
f_{Ackley}	m.	2.5×10^{-3}	4.8×10^{-4}	1.8×10^{-2}	9.2
$t = 1500$	σ	8.1×10^{-4}	0	2.1×10^{-3}	2.8
$f_{Griewank}$	m.	3.6×10^{-2}	5.8×10^{-2}	1.6×10^{-2}	8.6×10^{-2}
$t = 2000$	σ	3.2×10^{-2}	7.5×10^{-2}	2.2×10^{-2}	0.12
$f_{Rastrigin}$	m.	3.9×10^{-2}	18.7	4.6×10^{-2}	89.0
$t = 5000$	σ	1.9×10^{-1}	7.4	1.2×10^{-2}	23.1
$f_{Schwefel}$	m.	3.8×10^{-4}	216.04	14.987	4652.3
$t = 9000$	σ	3.0×10^{-9}	163.8	52.6	634.5
$f_{Rosenbrock}$	m.	11.73	7.18	5.06	6.17
$t = 20000$	σ	18.36	6.77	5.87	13.61

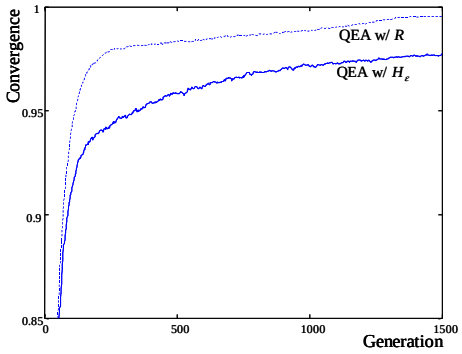
Table 5.2: Experimental results of the six numerical optimization functions of (A.1)-(A.6). The results of EP were referred from [80]. The number of runs was 50. The parenthesized values are the population sizes. $m.$, σ, and t represent the mean best, the standard deviation, and the maximum number of generations, respectively.



(a) Mean best (f_{Sphere})

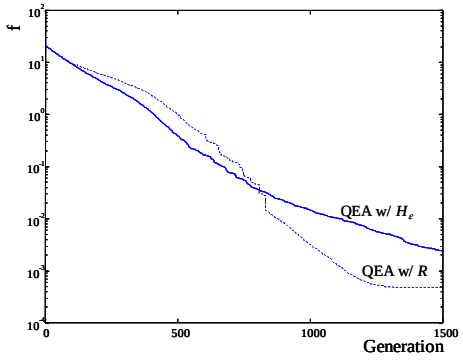


(b) Mean average (f_{Sphere})

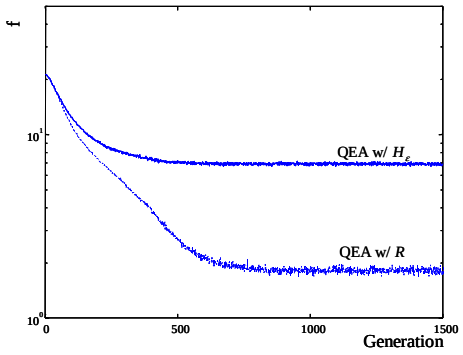


(c) Convergence (f_{Sphere})

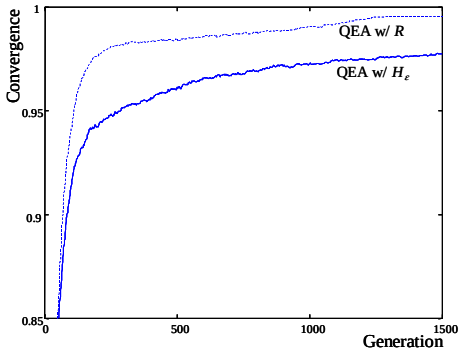
Figure 5.1: Comparison between QEA with the H_ϵ gate and QEA with the rotation gate on Sphere function. The parameter values were set to the same as shown in Table 5.1. The vertical axis for (a) and (b) shows the function value, the vertical axis for (c) shows the value of C_{av}, and the horizontal axis represents the number of generations. (a) shows the best result, (b) the average result, and (c) the average Q-bit convergence. All results were averaged over 50 runs.



(a) Mean best (f_{Ackley})

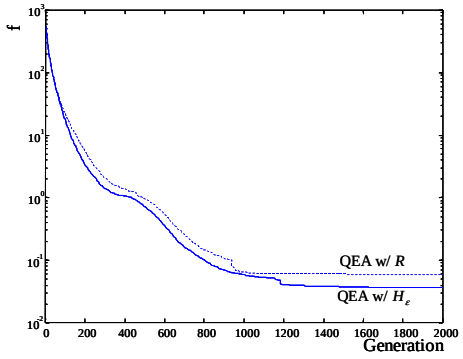


(b) Mean average (f_{Ackley})

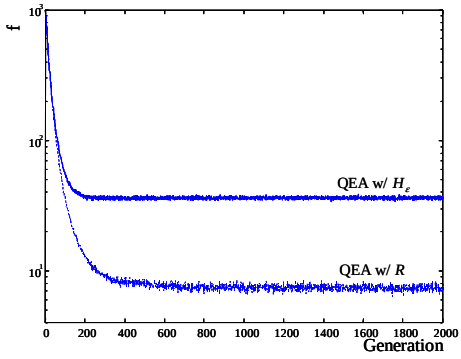


(c) Convergence (f_{Ackley})

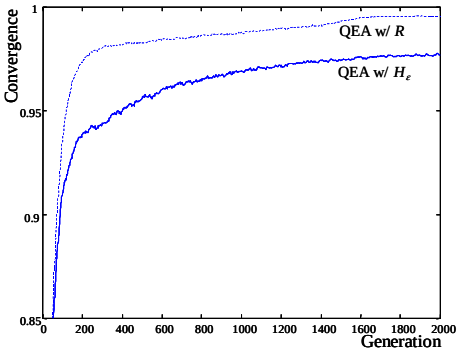
Figure 5.2: Comparison between QEA with the H_ϵ gate and QEA with the rotation gate on Ackley function. The parameter values were set to the same as shown in Table 5.1. The vertical axis for (a) and (b) shows the function value, the vertical axis for (c) shows the value of C_{av}, and the horizontal axis represents the number of generations. (a) shows the best result, (b) the average result, and (c) the average Q-bit convergence. All results were averaged over 50 runs.



(a) Mean best ($f_{Griewank}$)

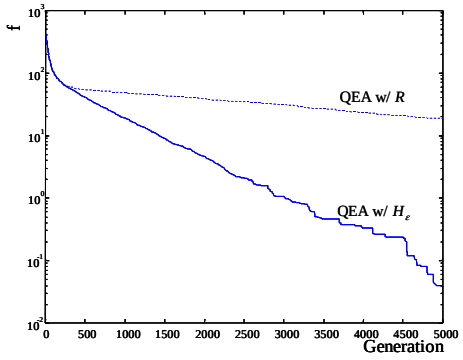


(b) Mean average ($f_{Griewank}$)

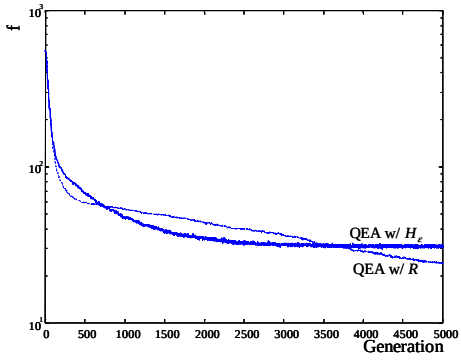


(c) Convergence ($f_{Griewank}$)

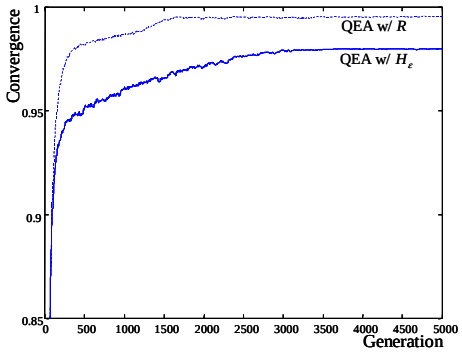
Figure 5.3: Comparison between QEA with the H_ϵ gate and QEA with the rotation gate on Griewank function. The parameter values were set to the same as shown in Table 5.1. The vertical axis for (a) and (b) shows the function value, the vertical axis for (c) shows the value of C_{av}, and the horizontal axis represents the number of generations. (a) shows the best result, (b) the average result, and (c) the average Q-bit convergence. All results were averaged over 50 runs.



(a) Mean best ($f_{Rastrigin}$)

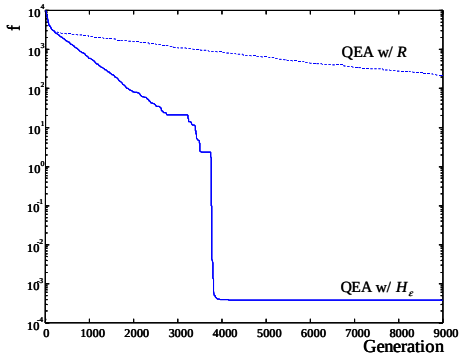


(b) Mean average ($f_{Rastrigin}$)

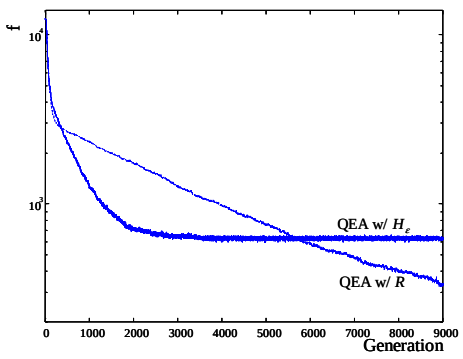


(c) Convergence ($f_{Rastrigin}$)

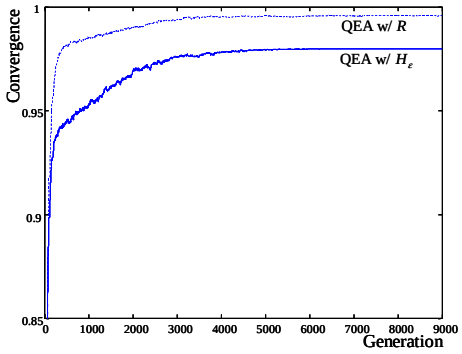
Figure 5.4: Comparison between QEA with the H_ϵ gate and QEA with the rotation gate on Rastrigin function. The parameter values were set to the same as shown in Table 5.1. The vertical axis for (a) and (b) shows the function value, the vertical axis for (c) shows the value of C_{av}, and the horizontal axis represents the number of generations. (a) shows the best result, (b) the average result, and (c) the average Q-bit convergence. All results were averaged over 50 runs.



(a) Mean best ($f_{Schwefel}$)



(b) Mean average ($f_{Schwefel}$)



(c) Convergence ($f_{Schwefel}$)

Figure 5.5: Comparison between QEA with the H_ϵ gate and QEA with the rotation gate on Schwefel function. The parameter values were set to the same as shown in Table 5.1. The vertical axis for (a) and (b) shows the function value, the vertical axis for (c) shows the value of C_{av}, and the horizontal axis represents the number of generations. (a) shows the best result, (b) the average result, and (c) the average Q-bit convergence. All results were averaged over 50 runs.

H_ϵ gate were somewhat worse compared to those of QEA with the rotation gate. The results of Griewank show, in particular, that the best results of QEA with the H_ϵ gate were better, although the average results of QEA with the H_ϵ gate were worse. The reason behind this type of result is the H_ϵ gate, that prevents each Q-bit converging to the final state (0 or 1). It is worthwhile to mention here that the results of the average Q-bit convergence show that the final values for QEAs with the H_ϵ and rotation gates converge to $(1 - 2\epsilon)$ and 1, respectively. Since the converged entropy of QEA with the H_ϵ gate is a nonzero value of (4.3), the converged Q-bit individual still includes various binary solutions. It means that the solutions selected as a population are obtained from a little bit wide area in the search space. From this reason, the average results of QEA with the H_ϵ gate did not converge below a certain value.

It is also worthwhile to mention that the QEA with the rotation gate is recommended for a class of unimodal functions which have no local optimum, and the QEA with the H_ϵ gate is recommended for a class of multimodal functions which have many local optima.

5.1.2 First hitting time

Three De Jong functions (see Appendix A.2) were considered to verify the performance of QEA with a single individual. The theoretical analysis of QEA with a single individual for the ONEMAX problem was already discussed in Section 3.6. For comparison purpose, simulated annealing (SA) was considered. The procedure SA is described in Appendix B.1. As a performance measure of the algorithms, we picked up the best search cost for the first hitting time over 50 runs. The number of times the fitness function was called was regarded as the search cost, since the evaluation of fitness function generally consumes of the most time compared to any other function. The number of bits for the three De Jong functions was set to 25 bits

		$f_{DeJong1}$	$f_{DeJong2}$	$f_{DeJong3}$
	m.	31544.9	9196.0	3802.1
QEA	σ	15944.0	1764.2	797.8
(0.0001π)	$r.$	50 / 50	50 / 50	50 / 50
	m.	122712.1	4117.2	1229.9
QEA	σ	94058.7	3962.7	344.5
(0.0005π)	$r.$	50 / 50	50 / 50	50 / 50
	m.	141143.2	7306.8	894.7
QEA	σ	90850.7	16085.5	248.4
(0.001π)	$r.$	50 / 50	50 / 50	50 / 50
	m.	299097.1	1705.0	1279.4
SA	σ	145643.4	826.3	782.0
(0.01)	$r.$	50 / 50	50 / 50	50 / 50
	m.	185057.8	1446.9	1207.5^*
SA	σ	71646.6	682.3	916.2^*
(0.1)	$r.$	50 / 50	50 / 50	46 / 50
	m.	193786.4	1446.8	706.2^*
SA	σ	64214.7	637.0	409.7^*
(1.0)	$r.$	50 / 50	50 / 50	21 / 50

Table 5.3: Experimental results of the three De Jong functions (A.7)-(A.9). Each parenthesized value of QEA is the rotation angle p (or $|n|$) and that of SA is the value of cooling parameter k for its temperature scheduler. The number of runs was 50. $m.$, σ, and $r.$ represent the mean best of search cost, the standard deviation of search cost, and the success rate, respectively. The values marked with * were obtained excluding the failure cases for which search cost was greater than 10^6.

(per variable). The value of ϵ for the H_ϵ gate was set to 0.01π.

Table 5.3 shows the experimental results for the three De Jong functions (A.7)-(A.9). In the results of $f_{DeJong1}$ and $f_{DeJong3}$, QEA with a single individual yielded better results compared to SA. In the results of $f_{DeJong2}$, which is relatively simple function compared to the other functions, SA performed better compared to QEA with a single individual. In particular, it should be noted that the results of $f_{DeJong3}$ showed that SA had several failure cases of which search cost was greater than 10^6. The reason is that De Jong function (3) has many discontinuous valleys as shown

in Figure A.2 (c) and SA may fall into one such valley. From these results, it is worthwhile to mention that QEA with a single individual performs better although the search space is distorted and it has many discontinuous valleys.

5.2 Combinatorial optimization

The experimental results of the knapsack problems with the average knapsack capacity were already discussed in Section 3.3. In this section, the knapsack problem with the restrictive knapsack capacity was considered for a class of combinatorial optimization problem to demonstrate the applicability of TPQEA. Comparison was made with the experimental results of QEA. The random repair method used in Section 3.3 was considered for handling the constraint of the knapsack capacity to compare their performance, although the greedy repair method guaranteed better solutions for the restrictive knapsack problem. Table 5.4 shows the parameter settings of TPQEA and QEA. The parameters of TPQEA were set to the same set of values as those of QEA except the additional parameters for the first phase of TPQEA. The rotation angle of p (or $|n|$) for Q-gate was set to 0.01π.

Table 5.5 shows the experimental results of the knapsack problems with 100, 250, and 500 items. As the table shows, TPQEA yielded much better results compared to QEA. Moveover, the average elapsed number of generations of TPQEA is smaller than that of QEA. More specifically, the convergence speed of TPQEA is 1.8, 2.9, and 4.0 times faster than that of QEA for 100, 250, and 500 items, respectively. The results of the standard deviations show that TPQEA is more robust than QEA for finding solutions.

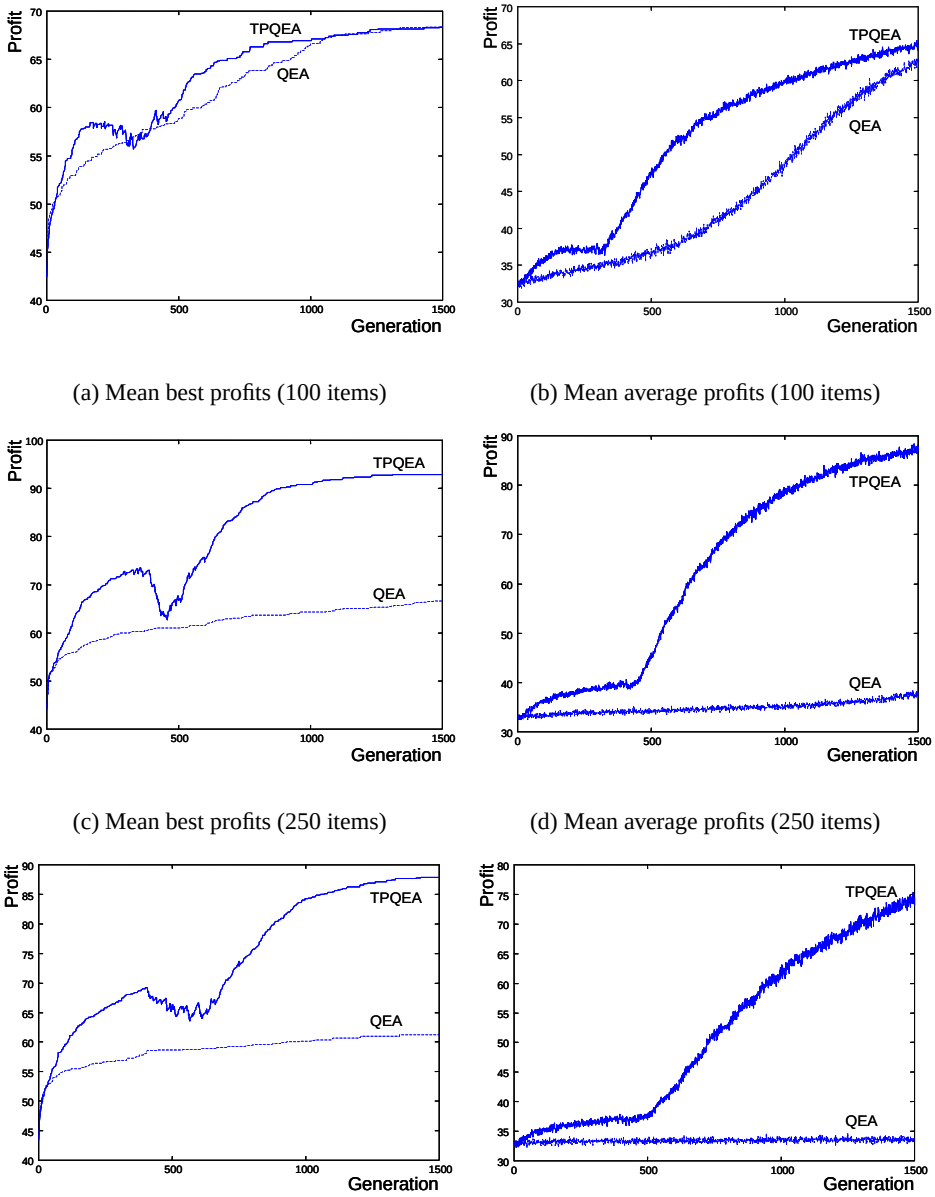
Figure 5.6 shows clearly that TPQEA performs significantly better than QEA in terms of convergence speed and the amount of profit. The transition point from the first phase to the second phase can be found out easily. After the transition, the rising slope of TPQEA is steeper than that of QEA. In particular, the tendency of

	QEA	TPQEA	
		Phase I	Phase II
Population size	15	15	
Number of observations	1	1	
Local group size	3	3	
Global migration period	100	-	100
Termination condition: γ	0.99	0.99	0.99
Equation of the termination condition	(3.19)	(3.20)	(3.19)
Minimum probability of (4.7): δ	-	0.01	-

Table 5.4: Parameter settings of QEA and TPQEA for the experiments on the restrictive knapsack problem. '-' means that the parameter is not needed.

		QEA	TPQEA
100	best	69.998	69.999
	mean	67.819	68.467
	worst	59.995	64.969
	σ	3.774	2.279
	t	1463	804
250	best	94.998	94.997
	mean	87.484	90.122
	worst	74.991	84.674
	σ	4.604	3.551
	t	2680	929
500	best	89.998	94.968
	mean	81.788	85.309
	worst	69.983	74.983
	σ	5.082	4.998
	t	4624	1165

Table 5.5: Experimental results of the knapsack problem with the restrictive knapsack capacity, $C = 20$, for 100, 250, and 500 items, respectively. The parameter values were set to the same as shown in Table 5.4, the number of runs was selected to be 30. σ and t represent the standard deviation and the average number of generations for termination, respectively.



(a) Mean best profits (100 items)

(b) Mean average profits (100 items)

(c) Mean best profits (250 items)

(d) Mean average profits (250 items)

(e) Mean best profits (500 items)

(f) Mean average profits (500 items)

Figure 5.6: Comparison of TPQEA and QEA on the restrictive knapsack problem. The parameter values were set to the same as shown in Table 5.4. However, regardless of the termination criteria, the results from 1 to 1,500 generations were plotted. The vertical axis shows the profit value of knapsack, and the horizontal axis represents the number of generations. The best profit values and the average profit values were averaged over 30 runs.

convergence rate is shown clearly in the results of the mean of average profits of population. After the transition, all the population converge to better solutions at a faster rate.

5.3 Summary

In this chapter, several numerical and combinatorial optimization problems were discussed to demonstrate the effectiveness and applicability of QEA. From the results of the six numerical optimization functions, the QEA with the rotation gate is recommended for a class of unimodal functions which have no local optimum, and the QEA with the H_ϵ gate is recommended for a class of multimodal functions which have many local optima. From the results of the three De Jong functions, the performance of QEA with a single individual was proved to be performed well although the search space is not simple. The experimental results of the restrictive knapsack problem also showed that TPQEA performed much better than QEA for a class of optimization problems with the global optimum which is present in the search space with larger ones-number distance from randomly generated search space.

6. Conclusions and Future Works

This thesis has proposed a novel quantum-inspired evolutionary algorithm (QEA) inspired by the concept of quantum computing. A Q-bit individual is defined to be a string of Q-bits for the probabilistic representation. To introduce the variation to the Q-bit individual, a new Q-gate variation operator is designed. The proposed QEA is characterized by the Q-bit representation for generating 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 set by the convergence of the Q-bit individuals.

The knapsack problem is considered to be an application appropriate enough to investigate the performance of QEA. The experimental results show that QEA performed quite well even with only one Q-bit individual. The characteristics of QEA could be verified by the simple knapsack problem with only 10 items. The results demonstrate the effectiveness and applicability of QEA to a class of combinatorial optimization problems.

The theoretical analysis of the QEA algorithm for the ONEMAX problem shows that QEA can maintain the balance between exploration and exploitation.

The definition of the Q-bit convergence could provide a meaningful termination criterion. By examining the effects of changing the values of the parameters of QEA, some guidance to the parameter settings could be introduced. In particular, some research issues for QEA such as the analysis of changing the initial values of Q-bits, a novel variation operator H_ϵ gate, and a two-phase QEA (TPQEA)

scheme are addressed to improve the performance of QEA. The experimental results of Schwefel function and the concatenated 5-bit traps show the effectiveness and applicability of the H_ϵ gate and TPQEA, respectively.

The results from several numerical optimization problems verify that QEA can be applied to a class of numerical as well as combinatorial optimization problems. In fact, the results have broken the conventional belief that a binary representation is not suitable for numerical optimization. These results extend the applicability and effectiveness of QEA for solving a general class of optimization problems.

Future research includes studying the dependencies among Q-bits as inspired by the concept of quantum entanglement to handle the dependencies among variables. For example, Rosenbrock function has 30 variables which have a strong dependency on each other. In this case, it is not easy to find the global optimum by using only a fitness function. A technique for handling the dependencies among variables will be useful to find the global optimum, although it may reduce the entropy of the search space. The technique can be implemented by considering the dependencies among Q-bits.

Appendix A. Optimization Problems

A.1 Knapsack problem

Knapsack problem which is a well-known combinatorial optimization problem is included in a class of NP-hard problems [81]. The knapsack problem can be described as selecting a subset of items from among various items so that it is 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 the condition

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

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 ith item is selected for the knapsack.

There are two types of knapsack capacity [68]:
i) average knapsack capacity

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

and ii) restrictive knapsack capacity $C = 2v$.

A.2 Numerical optimization problems

The following numerical optimization functions were considered in this thesis.

Sphere function: Minimize

$$f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2, \tag{A.1}$$

where $-100.0 \leq x_i \leq 100.0$ and $N = 30$. The global minimum value is 0.0 at $\mathbf{x} = (0, 0, \cdots, 0)$.

Ackley function: Minimize

$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) - \exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right) + 20 + e \tag{A.2}$$

where $-32.0 \leq x_i \leq 32.0$ and $N = 30$. The global minimum value is 0.0 at $\mathbf{x} = (0, 0, \cdots, 0)$.

Griewank function: Minimize

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{A.3}$$

where $-600.0 \leq x_i \leq 600.0$ and $N = 30$. The global minimum value is 0.0 at $\mathbf{x} = (0, 0, \cdots, 0)$.

Rastrigin function: Minimize

$$f(\mathbf{x}) = 10N + \sum_{i=1}^{N}\left(x_i^2 - 10\cos(2\pi x_i)\right), \tag{A.4}$$

where $-5.12 \leq x_i \leq 5.12$ and $N = 30$. The global minimum value is 0.0 at $\mathbf{x} = (0, 0, \cdots, 0)$.

Schwefel function: Minimize

$$f(\mathbf{x}) = 418.9829N - \sum_{i=1}^{N} x_i \sin(\sqrt{|x_i|}), \tag{A.5}$$

where $-500.0 \leq x_i \leq 500.0$ and $N = 30$. The global minimum value is 0.0 at $\mathbf{x} = (-420.9687, -420.9687, \cdots, -420.9687)$.

Rosenbrock function: Minimize

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right), \tag{A.6}$$

where $-30.0 \leq x_i \leq 30.0$ and $N = 30$. The global minimum value is 0.0 at $\mathbf{x} = (1, 1, \cdots, 1)$.

De Jong function (1): Minimize

$$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \tag{A.7}$$

where $-2.048 \leq x_i \leq 2.048$. The global minimum value is 0.0 at $(x_1, x_2) = (1, 1)$.

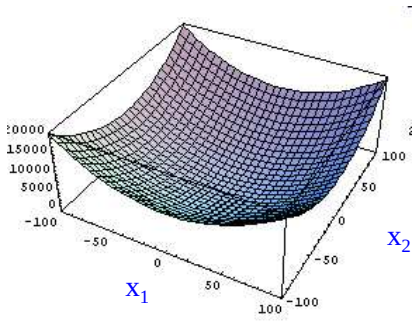
De Jong function (2): Minimize

$$f(\mathbf{x}) = \sum_{i=1}^{5} \text{integer}(x_i), \tag{A.8}$$

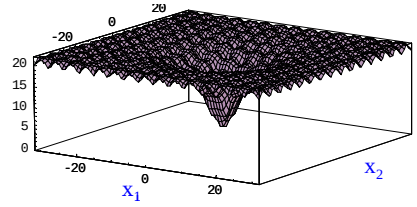
where $-5.12 \leq x_i \leq 5.12$. The global minimum value is -30 for all $-5.12 \leq x_i < -5.0$.

De Jong function (3): Minimize

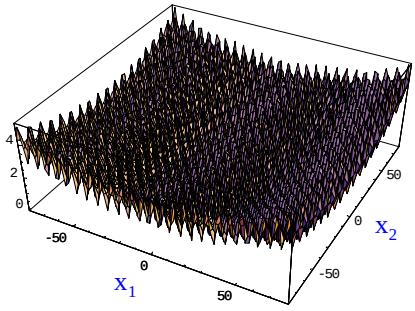
$$f(\mathbf{x}) = \frac{1}{\frac{1}{K} + \sum_{j=1}^{25} g_j^{-1}(x_1, x_2)}, \text{ where } g_j(x_1, x_2) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6, \tag{A.9}$$



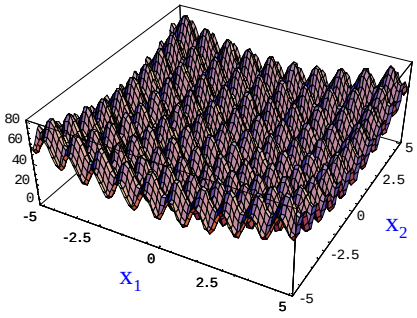
(a) Sphere function



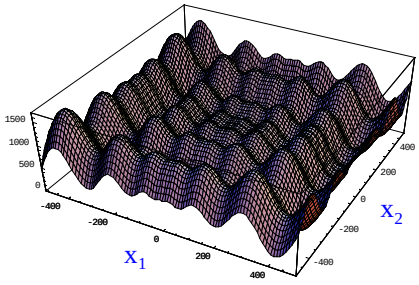
(b) Ackley function



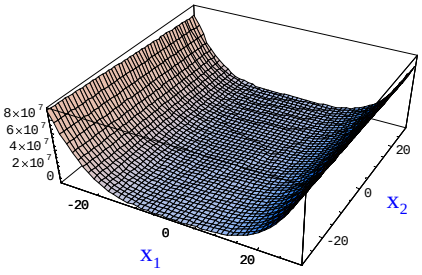
(c) Griewank function



(d) Rastrigin function

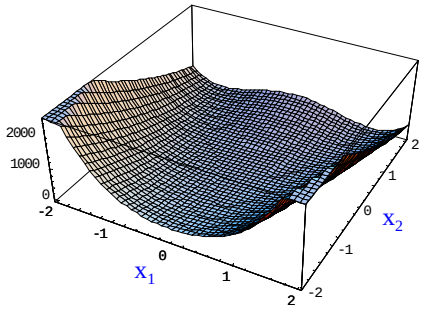


(e) Schwefel function

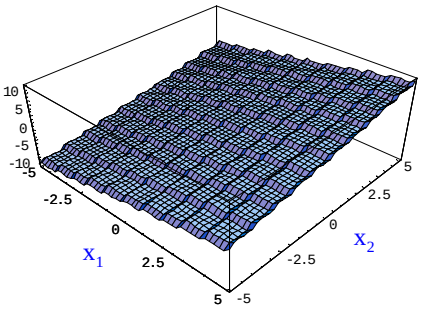


(f) Rosenbrock function

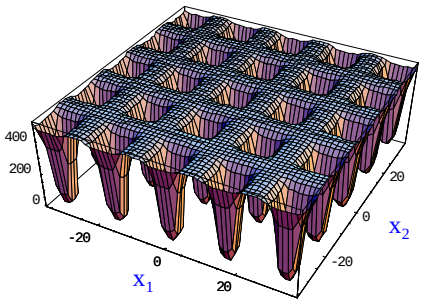
Figure A.1: Numerical optimization functions of (A.1)-(A.6). Each of them has 30 variables. In these figures, however, only two variables were considered to plot their shapes approximately.



(a) De Jong function (1)



(b) De Jong function (2)



(c) De Jong function (3)

Figure A.2: De Jong functions of (A.7)-(A.9).

where $-65.536 \leq x_i \leq 65.536$, $K = 500$, $c_j = j$, and

$$[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32 \end{bmatrix}.$$

The global minimum value is 0.998 at $(x_1, x_2) = (-32, -32)$.

Figures A.1 and A.2 show their shapes approximately.

A.3 Rosenbrock function

As shown in Table 5.2, no algorithm could find out the global minimum of Rosenbrock function, although the maximum number of generations was selected to be 20,000. In the experiments, we could find that x_{30} did not converge to the optimal value 1 in most of the experiments. In particular, the error of x_{30} induced the error of x_{29} and the error of x_{i+1} also induced the error of x_i.

Let us see the Rosenbrock function with respect to each pair of (x_i, x_{i+1}):

$$\begin{aligned} f(x_i, x_{i+1}) &= 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \\ &= f_e(x_i, x_{i+1}) + f_s(x_i). \end{aligned} \tag{A.10}$$

The function of $f(x_i, x_{i+1})$ can be divided into two functions, $f_e(x_i, x_{i+1})$ and $f_s(x_i)$ as shown in (A.10). The optimal value of x_i is determined by $f_s(x_i)$. However, since the coefficient of $(x_{i+1} - x_i^2)^2$ in $f_e(x_i, x_{i+1})$ is too big, $f_s(x_i)$ is negligible. From this reason, each x_i converges not to 1, but to $\sqrt{x_{i+1}}$.

Appendix B. Iterative Search Algorithms

B.1 Simulated annealing

Simulated annealing method is quite similar to the hill climbing method [79]. Instead of picking the best move, it picks a random move. If the move actually improves the situation, it is always executed. Otherwise, the algorithm makes the move with some probability less than 1. The probability decreases exponentially as time advances.

Figure B.1 shows the procedure SA which is a specific version for binary representation. In this figure, $\mathbf{x}_c$ is a current binary string, $\mathbf{x}_n$ a new binary string, T the current temperature, t the time step, $s(T,t)$ the scheduler for the temperature T, $f(\cdot)$ the fitness function of the problem, and $random[0,1]$ a random number from the range $[0,1)$.

There are several techniques for implementing the temperature scheduler $s(T,t)$. In this thesis, the following technique was used for implementing the scheduler $s(T,t)$:

$$s(T,t) = \frac{1}{k(t+1)}, \tag{B.1}$$

where k is the parameter for cooling temperature.

Procedure SA
begin
$t \leftarrow 0$
initialize temperature T
select a current string $\mathbf{x}_c$ at random
while ($T > 0$) **do**
begin
$t \leftarrow t + 1$
$T \leftarrow s(T, t)$
select a new string $\mathbf{x}_n$
in the neighborhood of $\mathbf{x}_c$
by flipping a single bit of $\mathbf{x}_c$
$\Delta E \leftarrow f(\mathbf{x}_n) - f(\mathbf{x}_c)$
if ($\Delta E > 0$)
then $\mathbf{x}_c \leftarrow \mathbf{x}_n$
else if ($\exp^{\Delta E/T} > random[0, 1]$)
then $\mathbf{x}_c \leftarrow \mathbf{x}_n$
end
end

Figure B.1: Simulated annealing.

요 약 문

양자 개념을 도입한 진화 알고리즘

본 논문에서는 새로운 진화 알고리즘인 양자 개념을 도입한 진화 알고리즘 QEA를 제안한다. QEA는 양자 비트나 상태의 중첩과 같은 양자 전산의 개념과 원리에 기반을 둔다. QEA는 다른 진화 알고리즘과 유사하게 개체의 표현법, 평가 함수, 개체들의 변화 등에 의해 특징지워진다. 하지만, QEA는 이진 표현법이나 실수 표현법 또는 기호 표현법 등이 아닌 확률적 표현법인 새로운 Q-비트 표현법을 정의하여 사용한다. Q-비트는 QEA에서의 정보의 최소 단위로 정의되며, Q-비트들로 이루어진 스트링은 Q-비트 개체로 정의된다. 이와 같이 확률적으로 존재하는 Q-비트 개체는 많은 이진 해들을 각기 다른 확률로 동시에 포함할 수 있게 된다. 이때 Q-비트 개체가 더 나은 해들을 높은 확률로 포함하도록 만들 수 있는 변형 연산자를 Q-게이트로 정의하고, 기본 Q-게이트로는 회전 게이트를 사용한다. 또한, Q-비트 수렴도를 정의함으로써 한 세대의 모든 Q-비트 개체들의 평균 Q-비트 수렴도를 이용하여 QEA의 명확한 종료 조건을 제시한다.

제안된 QEA 알고리즘의 특징을 파악하기 위해 간단한 문제에 대한 실험적, 이론적 분석을 수행한다. 특히, 이론적인 분석의 결과는 QEA 알고리즘이 진화 알고리즘에서 가장 중요하게 여겨지는 전역 검색과 지역 검색의 균형을 이루는데 적합한 구조를 가지고 있다는 사실을 입증한다. 또한, QEA의 파라메터들의 변화에 따른 영향을 분석함으로써 사용자에게 파라메터 설정에 대한 유용한 지침을 마련한다.

QEA의 성능을 더욱 향상시키기 위해 기본 알고리즘 구조의 확장에 대한 연구 주제를 추가적으로 다룬다. Q-비트의 초기 값 변화에 대한 영향을 분석하고, 초기 값에 대한 분석 결과를 바탕으로 2 단계 QEA 구조를 제안한다. 또한, 많은 국부 최적해를 갖는 수치 최적화 문제에 적합한 변형 연산자인 H_ϵ 게이트를 제안한다.

QEA의 효율성과 적용성을 검증하기 위해 조합 최적화의 대표적 문제인 주머니 문제와 다양한 수치 최적화 문제들에 대한 실험을 수행한다. 주머니 문제의 실험 결과는 QEA 알고리즘이 기존의 유전자 알고리즘에 비해 적은 수의 개체만으로도 조기 수렴 없이 우수한 성능을 보인다는 사실을 입증하였고, 수치 최적화 문제에 대한 실험 결과에서도 QEA 알고리즘의 우수한 성능이 입증되었다. 이와 같은 실험 결과는 QEA 알고리즘이 조합 최적화 뿐만 아니라 수치 최적화 문제에도 적용 가능한 진화 알고리즘이라는 사실을 입증해주고 있다.

References

[1] A. S. Fraser, "Simulation of Genetic Systems by Automatic Digital Computers," *Australian Journal of Biological Sciences*, vol. 10, pp. 484-491, 1957.

[2] H. J. Bremermann, "Optimization through Evolution and Recombination," in *Self-Organizing Systems*, eds. M. C. Yovits, G. T. Jacobi, and G. D. Goldstine, Washington D.C.: Spartan Books, pp. 93-106, 1962.

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

[4] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: John 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 Inter-Science, 1995.

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

[8] R. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467-488, 1982.

[9] D. Deutsch, "Quantum Theory, the Church-Turing principle and the universal quantum computer," in *Proceedings of the Royal Society of London A*, vol. 400, pp. 97-117, 1985.

[10] D. Deutsch, "Qautnum computational networks," in *Proceedings of the Royal Society of London A*, vol. 425, pp. 73-90, 1989.

[11] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," in *Proceedings of the Royal Society of London A*, vol. 439, pp. 553-558, 1992.

[12] D. R. Simon, “On the Power of Quantum Computation,” in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, Piscataway, NJ: IEEE Press, pp. 116-123, Nov. 1994.

[13] P. W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, Piscataway, NJ: IEEE Press, pp. 124-134, Nov. 1994.

[14] P. W. Shor, “Quantum Computing,” *Documenta Mathematica*, vol. Extra Volume ICM, pp. 467-486, 1998, http://east.camel.math.ca/ EMIS/ journals/ DMJDMV/ xvol-icm/ 00/ Shor.MAN.html.

[15] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the 28th ACM Symposium on Theory of Computing*, pp. 212-219, 1996.

[16] L. K. Grover, “Quantum Mechanics Helps in Searching for a Needle in a Haystack,” *Physical Review Letters*, American Physical Society, vol. 79, no. 2, pp. 325-328, Jul. 1997.

[17] L. K. Grover, “Quantum Mechanical Searching,” in *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 3, pp. 2255-2261, Jul. 1999.

[18] L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy, “Finding a Better-than-Classical Quantum AND/OR Algorithm using Genetic Programming,” in *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 3, pp. 2239-2246, Jul. 1999.

[19] S.-C. Lee, *Quantum Computation*. Technical report, Department of Physics, Korea Advanced Institute of Science and Technology, Korea.

[20] R. L. Rivest, A. Shamir, and L. Adleman, “A method of obtaining digital signatures and public-keycryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[21] C. P. Williams and S. H. Clearwater, *Explorations in Quantum Computing*. Berlin: Springer-Verlag, 1998.

[22] B. I. P. Rubinstein, “Evolving Quantum Circuits using Genetic Programming,” in *Proceedings of the 2001 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 1, pp. 144-151, May 2001.

[23] M. Lukac and M. Perkowski, "Evolving Quantum Circuits using Genetic Algorithm," in *Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware*, Piscataway, NJ: IEEE Press, pp. 177-185, Jul. 2002.

[24] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, pp. 61-66, May 1996.

[25] K.-H. Han and J.-H. Kim, "Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem," in *Proceedings of the 2000 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 2, pp. 1354-1360, Jul. 2000.

[26] K.-H. Han, K.-H. Park, C.-H. Lee, and J.-H. Kim, "Parallel Quantum-inspired Genetic Algorithm for Combinatorial Optimization Problem," in *Proceedings of the 2001 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 2, pp. 1422-1429, May 2001.

[27] K.-H. Han and J.-H. Kim, "Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization," *IEEE Transactions on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 6, no. 6, pp. 580-593, Dec. 2002.

[28] M. Moore and A. Narayanan, *Quantum-inspired Computing*. Technical report, Department of Computer Science, University of Exeter, UK, 1995.

[29] K.-H. Kim, J.-Y. Hwang, K.-H. Han, J.-H. Kim, and K.-H. Park, "A Quantum-inspired Evolutionary Computing Algorithm for Disk Allocation Method," *IEICE Transactions on Information and Systems*, IEICE Press, vol. E86-D, no. 3, pp. 645-649, Mar. 2003.

[30] J.-S. Jang, K.-H. Han, and J.-H. Kim, "Quantum-inspired Evolutionary Algorithm-based Face Verification," accepted for publication in *Proceedings of the Genetic and Evolutionary Computation Conference*, AAAI Press, Jul. 2003.

[31] AR face database: http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html

[32] T. Menneer and A. Narayanan, *Quantum-inspired Neural Networks*. Technical report, Department of Computer Science, University of Exeter, UK, 1995.

[33] S. B. Garavaglia, "A Quantum-inspired Self-Organizing Map," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, Piscataway, NJ: IEEE Press, vol. 2, pp. 1779-1784, 2002.

[34] M. Pelikan, D. E. Goldberg, and F. Lobo, *A Survey of Optimization by Building and Using Probabilistic Models*. Technical report: IlliGAL Report No. 99018, Department of General Engineering, University of Illinois at Urbana-Champaign, IL, Sep. 1999.

[35] S. Baluja, *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Technical report: CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, PA, Jun. 1994.

[36] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," in *Proceedings of the International Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, pp. 523-528, May 1998.

[37] G. R. Harik, *Linkage Learning via Probabilistic Modeling in the ECGA*. Technical report: IlliGAL Report No. 99010, Department of General Engineering, University of Illinois at Urbana-Champaign, IL, Jan. 1999.

[38] M. Pelikan, *Bayesian Optimization Algorithm: From Single Level to Hierarchy*. Ph.D. dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, US, 2002.

[39] G. W. Greenwood, "Finding Solutions to NP Problems: Philosophical Differences Between Quantum and Evolutionary Search Algorithms," in *Proceedings of the 2001 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, pp. 815-822, May 2001.

[40] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd ed. Piscataway, NJ: IEEE Press, 2000.

[41] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[42] T. Bäck, "Evolutionary Algorithms and Their Standard Instances: Introduction," in *Handbook of Evolutionary Computation*, eds. T. Bäck, D. B. Fogel, and Z. Michalewicz, New York: Oxford University Press, B1.1:1-B1.1:4, 1997.

[43] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *Computer*, Piscataway, NJ: IEEE Press, vol. 27, no. 6, pp. 17-26, Jun. 1994.

[44] J. R. Koza, "Survey of Genetic Algorithms and Genetic Programming," in *Proceedings of the 1995 WESCON conference*, Piscataway, NJ: IEEE Press, pp. 589-594, Nov. 1995.

[45] L. J. Eshelman, “Genetic Algorithms,” in *Handbook of Evolutionary Computation*, eds. T. Bäck, D. B. Fogel, and Z. Michalewicz, New York: Oxford University Press, B1.2:1-B1.2:11, 1997.

[46] G. Rudolph, “Evolution Strategies,” in *Handbook of Evolutionary Computation*, eds. T. Bäck, D. B. Fogel, and Z. Michalewicz, New York: Oxford University Press, B1.3:1-B1.3:6, 1997.

[47] V. W. Porto, “Evolutionary Programming,” in *Handbook of Evolutionary Computation*, eds. T. Bäck, D. B. Fogel, and Z. Michalewicz, New York: Oxford University Press, B1.4:1-B1.4:10, 1997.

[48] H. Myung, *Evolutionary Optimization Based on Lagrangian (Evolian) for Nonlinear Constrained Optimization Problems*. Ph.D. dissertation, Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea, 1998.

[49] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin “Experimental Quantum Cryptography,” *Journal of Cryptography*, Berlin: Springer-Verlag, vol. 5, pp. 3-28, 1992.

[50] C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. K. Wootters, “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels,” *Physical Review Letters*, American Physical Society, vol. 70, no. 13, pp. 1895-1899, Mar. 1993.

[51] A. M. Turing, “On Computable Numbers, with an Application to the Entsheidungsproblem,” in *Proceedings of the London Mathematical Society*, series 2, 42:230-265, 1936.

[52] R. Landauer, “Irreversibility and heat generation in the computing process,” *IBM Journal of Research and Development*, vol. 5, p. 183, 1961.

[53] C. H. Bennett, “Logical reversibility of computation,” *IBM Journal of Research and Development*, vol. 17, p. 525, 1973.

[54] P. Benioff, “Quantum Mechanical Models of Turing Machines That Dissipate No Energy,” *Physical Review Letters*, American Physical Society, vol. 48, no. 23, pp. 1581-1585, Jun. 1982.

[55] T. Hey, “Quantum computing: an introduction,” *Computing & Control Engineering Journal*, Piscataway, NJ: IEEE Press, vol. 10, no. 3, pp. 105-112, Jun. 1999.

[56] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[57] http:// mrm.kaist.ac.kr/ qc/ basic/ qcom.html

[58] A. O. Pittenger, *An Introduction to Quantum Computing Algorithms*. Boston: Birkhäuser, 2000.

[59] A. Narayanan, "Quantum computing for beginners," in *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 3, pp. 2231-2238, Jul. 1999.

[60] J. Preskill, *Lecture Notes for Physics 229: Quantum Information and Computation*. Department of Physics, California Institute of Technology, Sep 1998. http:// www.theory.caltech.edu/ ∼preskill/ ph229.

[61] R. Hinterding, "Representation, Constraint Satisfaction and the Knapsack Problem," in *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 2, pp. 1286-1292, Jul. 1999.

[62] A. L. Olsen, "Penalty Functions and the Knapsack Problem," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 2, pp. 554-558, Jun. 1994.

[63] R. Spillman, "Solving Large Knapsack Problems with a Genetic Algorithm," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Piscataway, NJ: IEEE Press, vol. 1, pp. 632-637, Oct. 1995.

[64] G. R. Raidl, "An Improved Genetic Algorithm for the Multiconstrained 0-1 Knapsack Problem," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 1, pp. 207-211, May 1998.

[65] G. R. Raidl, "Weight-codings in a genetic algorithm for the multi-constraint knapsack problem," in *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 1, pp. 596-603, Jul. 1999.

[66] S. Ku and B. Lee, "A Set-Oriented Genetic Algorithm and the Kanpsack Problem," in *Proceedings of the 2001 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 1, pp. 650-654, May 2001.

[67] H. Handa, T. Horiuchi, O. Katai, T. Kaneko, T. Konishi, and M. Baba, "Coevolutionary GA with Schema Extraction by Machine Learning Techniques and Its Application to Knapsack Problems," in *Proceedings of the 2001 Congress*

on Evolutionary Computation, Piscataway, NJ: IEEE Press, vol. 2, pp. 1213-1219, May 2001.

[68] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd, revised and extended ed. Berlin: Springer-Verlag, 1999.

[69] J.-H. Kim and H. Myung, "Evolutionary Programming Techniques for Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 1, no. 2, pp. 129-140, Jul. 1997.

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

[71] G. Rudolph, "Convergence Rates of Evolutionary Algorithms for a Class of Convex Objective Functions," *Control and Cybernetics*, Poland: Systems Research Institute, vol. 26, no. 3, pp. 375-390, 1997.

[72] J. Garnier, L. Kallel, and M. Schoenauer, "Rigorous Hitting Times for Binary Mutations," *Evlutionary Computation*, MIT Press, vol. 7, no. 2, pp. 173-203, 1999.

[73] J. Garnier and L. Kallel, "Statistical Distribution of the Convergence Time of Evolutionary Algorithms for Long-Path Problems," *IEEE Transactions on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 4, no. 1, pp. 16-30, Apr. 2000.

[74] J. He and X. Yao, "Drift Analysis and Average Time Complexity of Evolutionary Algorithms," *Artificial Intelligence*, Elsevier Science, vol. 127, no. 1, pp. 57-85, Mar. 2001.

[75] S. Droste, T. Jansen, and I. Wegener,, "On the Analysis of the (1+1) Evolutionary Algorithm," *Theoretical Computer Science*, Elserier Science, vol. 276, pp. 51-81, Apr. 2002.

[76] J. He and X. Yao, "From an Individual to a Population: An Analysis of the First Hitting Time of Population-Based Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 6, no. 5, pp. 495-511, Oct. 2002.

[77] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Boston: IRWIN, 1993.

[78] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[79] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall, 1995.

[80] X. Yao, Y. Liu, and G. Lin, “Evolutionary Programming Made Faster,” *IEEE Transactions on Evolutionary Computation*, Piscataway, NJ: IEEE Press, vol. 3, no. 2, pp. 82-102, Jul. 1999.

[81] M. R. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman & Company, 1979.

감 사 의 글

이 논문이 완성되기까지 도움을 주신 많은 분들께 진심으로 감사를 드립니다. 먼저 오랜 기간동안 언제나 곁에서 지켜봐 주시고 값진 조언과 격려를 아끼지 않으신 김종환 교수님께 깊은 감사를 드립니다. 바쁜 일정에도 불구하고 많은 관심으로 논문을 심사해 주시고 조언을 아끼지 않으신 박규호 교수님, 임종태 교수님, 탁민제 교수님, 그리고 홍성철 교수님께 진심으로 감사를 드립니다.

오랜 기간 동안 연구실 생활을 하면서 많은 힘이 되어준 연구실원들께 감사의 마음을 전합니다. 많은 것을 깨닫게 해준 실험실의 선배님 정열이형, 현식이형, 광춘이형, 현이형, 정민이형, 신이형, 동균이형, 치호형, 명진이형, 홍수형, 동한이형, 용재형, 인환이형, 영원한 동기 귀홍, 상호, 광희, 그리고 문수형과 이진이형을 포함한 성실한 후배 승은, 밍, 강희, 준수, 재호, 용덕, 춘경, 훈봉, 기표, 범주, 윤기, 미회 모두에게 고마운 마음을 전합니다. 그리고, 짧은 만남이었지만 따뜻한 격려를 해주신 유지환 박사님, 논문 교정을 도와주신 Seow 박사님과 Ghoshal 박사님께도 감사를 드립니다.

그리고 멀리에서나 가까이에서나 항상 힘이 되어 주었던 친구들에게도 진심으로 고마운 마음을 전합니다.

일을 핑계로 단 하루도 마음 편하게 해주지 못했지만 언제나 사랑으로 함께 고민해주고, 격려해주고, 챙겨준 아내 명진에게 사랑과 고마운 마음을 전합니다. 그리고, 얼굴도 제대로 보여주지 못한 아빠를 항상 환한 미소로 맞아주는 딸 세라에게도 고맙다는 말을 전합니다.

마지막으로 저를 위해 항상 기도해주시고 아껴주시는 사랑하는 부모님과 먼 곳에서도 큰 힘이 되어준 누나, 형, 그리고 부족한 저를 항상 대견하게 여겨주시는 장인 장모님께 무한한 감사를 드립니다.

이 력 서

성 명	한 국 현
생 년 월 일	1975년 7월 11일
출 생 지	서 울
본 적	대전시 유성구 전민동
전 자 우 편	khhan@khhan.com
홈 페 이 지	http://www.khhan.com

학 력

1993. 3. ~ 1997. 2.	한국과학기술원 전기 및 전자공학과 (B.S.)
1997. 3. ~ 1999. 2.	한국과학기술원 전기 및 전자공학과 (M.S.)
1999. 3. ~ 2003. 8.	한국과학기술원 전자전산학과 (Ph.D.)

경 력

1998. 1. ~ 2000. 12.	SMT 생산 지원 시스템을 위한 작업 최적화 및 라인제안 프로그램 개발, 미래산업
1998. 9. ~ 2000. 8.	인터넷 기반 퍼스널 로봇 개발, 정보통신부
1999. 12. ~ 2000. 11.	초고속 병렬 진화 연산 S/W 및 응용 기술 개발, 과학기술부
2000. 4. ~ 2000. 12.	USB HUB-LAN 장치 제품 개발 및 양산 지원, 위더스/이노컴
2000. 9. ~ 2003. 8.	Stand alone type의 Vision board를 이용한 Human robot interaction (IEEE1394 Stereo Camera 개발), KIST
2001. 10. ~ 2003. 7.	퍼스널 로봇을 위한 정보/지능 기술 개발, 산업자원부

수 상

1997. 6. 5. 2nd Prize in S-MiroSot category at the 1st Micro-Robot World Cup Soccer Tournament, organized by KAIST, Korea.

2001. 2. 19. Bronze Prize at the 7th Samsung Humantech Thesis Prize Awards, organized by Samsung Electronics Co., Korea.

2001. 8. 5 2nd Prize in Middle League MiroSot category at the FIRA Cup China 2001, organized by FIRA, China.

2003. 2. 19. Gold Prize at the 9th Samsung Humantech Thesis Prize Awards, organized by Samsung Electronics Co., Korea.

특 허

2001. 11. 23. "인터넷을 이용한 원격 이동로봇의 인터넷 직접제어 시스템 및 그 방법," 특허등록 제 0316630호, 한국.

2002. 6. 17. "로봇 축구 게임 방법," 특허등록 제 0342432호, 한국.

2002. 8. 13. "양자연산 개념을 도입한 진화연산방법," 특허등록 제 0350233호, 한국.

2002. 10. 31. "로봇 축구 게임 장치," 특허등록 제 0361031호, 한국.

Publications

[1] **Kuk-Hyun Han**, Sinn Kim, Yong-Jae Kim, and Jong-Hwan Kim, "Internet Control Architecture for Internet-Based Personal Robot," *Autonomous Robots Journal*, Kluwer Academic Publishers, Vol. 10, No. 2, pp. 135-147, Mar. 2001.

[2] **Kuk-Hyun Han** and Jong-Hwan Kim, "Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization," *IEEE Trans. on Evolutionary Computation*, IEEE Press, Vol. 6, No. 6, pp. 580-593, Dec. 2002.

[3] Kyung-Ho Kim, Joo-Young Hwang, **Kuk-Hyun Han**, Jong-Hwan Kim, and Kyu-Ho Park, "A Quantum-inspired Evolutionary Computing Algorithm for Disk Allocation Method," *IEICE Trans. on Information and Systems*, IEICE Press, Vol. E86-D, No. 3, pp. 645-649, Mar. 2003.

[4] **Kuk-Hyun Han** and Jong-Hwan Kim, "Quantum-inspired Evolutionary Algorithms with a New Termination Criterion, H_ϵ Gate, and Two-Phase Scheme," submitted to *IEEE Trans. on Evolutionary Computation*, IEEE Press, May 2003.

[5] **Kuk-Hyun Han** and Jong-Hwan Kim, "Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem," in *Proc. of the 2000 Congress on Evolutionary Computation*, IEEE Press, pp. 1354-1360, Jul. 2000.

[6] Jong-Hwan Kim, **Kuk-Hyun Han**, Shin Kim, and Yong-Jae Kim, "Internet-Based Personal Robot System using Map-Based Localization," in *Proc. of the 32nd Int. Symp. on Robotics*, pp. 1282-1287, Apr. 2001.

[7] **Kuk-Hyun Han**, Kui-Hong Park, Chi-Ho Lee, and Jong-Hwan Kim, "Parallel Quantum-inspired Genetic Algorithm for Combinatorial Optimization Problem," in *Proc. of the 2001 Congress on Evolutionary Computation*, IEEE Press, pp. 1422-1429, May 2001.

[8] **Kuk-Hyun Han**, Shin Kim, Yong-Jae Kim, Seong-Eun Lee, and Jong-Hwan Kim, "Implementation of Internet-Based Personal Robot with Internet Control Architecture," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, IEEE Press, pp. 217-222, May 2001.

[9] **Kuk-Hyun Han** and Jong-Hwan Kim, "Analysis of Quantum-inspired Evolutionary Algorithm," in *Proc. of the 2001 Int. Conf. on Artificial Intelligence*, CSREA Press, Vol. 2, pp. 727-730, Jun. 2001.

[10] **Kuk-Hyun Han**, Kang-Hee Lee, Choon-Kyoung Moon, Hoon-Bong Lee, and Jong-Hwan Kim, "Robot Soccer System of SOTY 5 for Middle League MiroSot," in *Proc. of the 2002 FIRA Robot World Congress*, pp. 632-635, May 2002.

[11] **Kuk-Hyun Han** and Jong-Hwan Kim, "Direct Internet Control Architecture for Personal Robot," in *Proc. of the 2002 FIRA Robot World Congress*, pp. 264-268, May 2002.

[12] **Kuk-Hyun Han** and Jong-Hwan Kim, "Introduction of Quantum-inspired Evolutionary Algorithm," in *Proc. of the 2002 FIRA Robot World Congress*, pp. 243-248, May 2002.

[13] **Kuk-Hyun Han**, Yong-Jae Kim, Jong-Hwan Kim, and Steve Hsia, "Internet Control of Personal Robot between KAIST and UC Davis," in *Proc. of the*

IEEE Int. Conf. on Robotics and Automation, IEEE Press, pp. 2184-2189, May 2002.

[14] Jun-Su Jang, **Kuk-Hyun Han**, and Jong-Hwan Kim, "Quantum-inspired Evolutionary Algorithm-based Face Verification," accepted for publication in *Proc. of the Genetic and Evolutionary Computation Conference*, AAAI Press, Jul. 2003.

[15] **Kuk-Hyun Han** and Jong-Hwan Kim, "On Setting the Parameters of QEA for Practical Applications: Some Guidelines based on Empirical Evidence," accepted for publication in *Proc. of the Genetic and Evolutionary Computation Conference*, AAAI Press, Jul. 2003.