양자 진화 알고리즘

한 국 현 (한국과학기술원 전자전산학과)

Abstract- This paper proposes a new evolutionary algorithm called quantum-inspired evolutionary algorithm(QEA). Quantum-inspired evolutionary algorithm, that is, QEA is based on the concept and principles of quantum computing such as quantum bit and a linear superposition of states. Instead of binary, numeric, or symbolic representation, by adopting quantum individual as a representation, QEA can represent a linear superposition of solutions due to its probabilistic representation. Rapid convergence and good global search capability characterize the performance of QEA. Especially, QEA is suitable for combinatorial optimization problem. The effectiveness and the applicability of QEA are demonstrated by experimental results on the knapsack problem, which is a well-known combinatorial optimization problem. The results show that QEA is superior to other conventional genetic algorithms.

I. 서론

진화 알고리즘은 기본적으로 자연의 생물학적 진화 원리에 기반하여 확률적인 검색과 최적화를 수행하는 방법이다. 계산에 의한 전통적인 최적화 방법과 비교하면, 진화 알고리즘은 강인성과 어떠한 문제에도 쉽게 적용될 수 있는 범용성을 갖는다. 진화 알고리즘은 이론적 배경이 부족하다는 사실에도 불구하고, 실험적 결과들은 일반적으로 기대 이상을 보여주고 있다. 진화 알고리즘은 수식적 연산으로 최적 해를 찾기 어려운 문제들에 보편적으로 적용되고 있다. 많은 사람들이 최적화 도구로 사용하고 있는 진화 알고리즘에는 크게 유전자 알고리즘(GA: genetic algorithm), 진화 프로그래밍(EP: evolutionary programming), 진화 전략(ES: evolution strategies) 등이 있다. 해의 표현법이나 진화 연산자 등에 있어서 약간의 차이점은 있지만, 세 가지 알고리즘의 기본 원리와 개념 등은 매우

흡사하다. 문제의 정확한 최적 해를 구하기 위해서는 가능한 모든 입력에 대한 연산을 수행해 보아야 한다. 하지만 대부분의 문제에 있어서 입력 영역이 너무 크기 때문에, 모든 연산을 수행한다는 것은 불가능하다. 이때 일부 연산만으로 빠른 시간 내에 최적에 근접한 해를 찾기 위해 진화 알고리즘이 사용되고 있다.

진화 알고리즘의 탐색에 있어서 가장 중요한 요소 중의 하나는 전역탐색(exploration)과 지역탐색(exploitation)의 균형이다. 전역탐색만을 수행할 경우에는 무작위 검색과 같은 결과를 나타내며, 지역탐색만을 수행할 경우에는 국부 최적 해를 찾는 결과를 초래하게 된다. 진화 알고리즘의 구현 측면에서 보면, 개체 집단의 다양성과 다음 세대의 선택 강도가 각각 전역탐색과 지역탐색에 해당한다. 선택 강도의 증가는 개체집단의 다양성을 감소시키고, 선택 강도의 감소는 반대로 개체 집단의 다양성을 증가시킨다[1]. 따라서 검색 초기에는 개체 집단의 다양성을 강조하고, 검색이 진행될수록 점차로 선택강도를 높여 지역탐색을 강조해야 한다. 하지만 검색 조절의 적합한 시점을 판단하는 것은 쉽지 않다.

진화 알고리즘, 특히 유전자 알고리즘(GA)이 왜 동작하는가에 대해서는 대부분 스키마(schema) 개념을 이용하여 설명한다[1]. 스키마는 유전 염색체들 사이의 유사성을 나타낼 수 있는 하나의 전형이다. 염색체의 우열을 나타내는 스키마가 존재한다는 것이다. 1001110001 염색체가 존재할 경우, 예를 들면 스키마 111**01이 염색체의 우열을 결정할 수 있다는 것이다. 진화 알고리즘에서는 세대를 거듭할수록 우성을 표현하는 스키마는 점점 더 많은 개체에 포함되게 된다는 것이다. 이때 우성을 표현하는 스키마가 존재한다는 사실은 알지만, 과연 어떤 것이 우성을 표현하는 스키마인가를 판단하는 것은 결코 쉽지 않다. 이는 현재까지 진화해온 개체들의 내력이 남아있지 않기 때문이다. 즉, 스키마의 우열 여부를 판단할 수 있는 정보는 부모 세대와 현재 세대의 개체들이 전부이다. 존재했던 모든 개체들의 정보를 남기기 위해서는 너무나 많은 저장 공간과 연산이 필요하다.

본 논문에서는 앞서 언급한 진화 알고리즘에서의 두 가지 문제점, 즉 전역탐색과 지역탐색의 균형 문제 및 진화해온 개체들의 내력의 부재로 우성 스키마 판단이 곤란한 문제 등을 해결할 수 있는 새로운 진화 알고리즘인 양자 진화 알고리즘(QEA: quantum-inspired evolutionary algorithm)을 제안한다. QEA는 양자 비트, 상태의 중첩 등과 같은 양자 연산 원리들의 개념을 이용한다. QEA 알고리즘은 이진 표현법, 숫자 표현법, 기호 표현법 등을 사용하지 않고 확률적인 표현법을 사용한다. 즉, 양자의 확률적 표현법을 이용함으로써 개체들의 내력을 모두 저장할 수 있으며, 초기 전역탐색에서 최적 해로 수렴함에 따라 지역탐색을 수행하는 기능을 갖고 있다. 이외에도 QEA는 다음과 같은 장점을 갖는다: 알

고리즘의 종료 조건을 결정할 수 있는 지표 값이 명확하다. 많은 개체를 동시에 표현할 수 있는 특징에 의해 적은 개체만을 이용하여 우수한 해를 얻을 수 있다. 적은 개체수로 인해 짧은 시간 내에 우수한 해를 얻을 수 있다.

QEA는 양자 연산 개념을 이용하여 구현한 진화 알고리즘이다. 양자 연산은 주로 양자 컴퓨터를 위한 양자 알고리즘을 연구하는 분야이다. 양자 컴퓨터는 1980년대 초반에 제안되었고[2], 1980년대 말에 구체화되었다[3]. 그 후 1990년대 초반부터 기존 컴퓨터보다 성능이 뛰어난 양자 컴퓨터에 대한 많은 연구가 진행되어 왔다. 참고로 현재까지 알려진 우수한 양자 알고리즘에는 쇼어의 소인수 분해 알고리즘[4,5]과 그루버의 데이터베이스 검색 알고리즘[6,7] 등이 있다. 1990년대 후반부터 몇몇 연구원들은 기존의 진화 연산 기법과 양자 연산 기법을 접목시키는 연구를 시작하였다. 두 기법을 접목시키는 연구는 크게 두 가지로 나누어 볼 수 있다. 그 중의 한가지는 새로운 양자 알고리즘을 유전자 프로그래밍(genetic programming)과 같은 자동 프로그래밍 기법(automatic programming technique)을 이용하여 생성해내는 연구이다[8]. 다른 한가지는 진화 연산 기법에 대한 연구의 일종으로, 기존의 컴퓨터에서 사용할 수 있는 양자 역학의 개념을 도입한 진화 연산 알고리즘의 연구이다[9]. 유전자 알고리즘의 교배 연산자에 간섭 개념을 추가한 정도의 간단한 알고리즘을 제안했다.

본 논문은 다음과 같이 구성된다. II장에서는 새로운 진화 알고리즘인 양자 진화 알고리즘(QEA)에 대해서 설명하고 특징을 분석한다. III장에서는 QEA의 구현 예를 보이기 위해 배낭(knapsack) 문제를 도입한다. 이때 배낭 문제 실험에서 사용된 기존의 여러 유전자 알고리즘과 구체적인 QEA 알고리즘에 대해 기술한다. IV장에서는 실험결과를 정리하고 분석한다. 마지막으로 V장에서 결론을 맺는다.

II. 양자 진화 알고리즘 (QEA: quantum-inspired evolutionary algorithm)

QEA는 양자 비트와 양자 역학의 상태의 중첩 개념 등을 기반으로 한다. 양자 컴퓨터에서 정보를 나타내는 기본이 되는 가장 작은 단위를 양자 비트(quantum bit)라고 한다 [10]. 양자 비트는 '1'의 상태나 '0'의 상태 또는 '0'과 '1'의 임의의 중첩 상태를 표현할 수 있다. 양자 비트의 상태를 표현하면 다음과 같다.

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1)$$

이때 α와 β는 '0'과 '1'의 상태의 확률 크기를 구체화하는 복소수이다. 양자 비트가 '0'의 상태에 있을 확률은 $|\alpha|^2$이고, '1'의 상태에 있을 확률은 $|\beta|^2$이 된다. 상태의 정규화에

의해 다음과 같은 조건을 보장한다.

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

만약 m개의 양자 비트 시스템이 있다면, 그 시스템은 2^m 가지의 상태를 동시에 표현할 수 있게 된다. 그러나 양자 상태를 관측하는 순간에는 많은 상태 중에서 한 가지 상태만으로 관측되게 된다. 본 장에서는 QEA를 위한 새로운 표현법을 기술하고, QEA 알고리즘을 제안한다.

1. 표현법 (Representation)

진화 알고리즘에서 해를 코딩하기 위한 표현법은 여러 가지가 있다. 기존의 표현법은 크게 3가지로 나누어 생각할 수 있다: 이진(binary) 표현법, 숫자(numeric) 표현법, 기호(symbolic) 표현법이 그것이다[11]. QEA에서는 양자 비트의 개념을 바탕으로 새로운 표현법을 이용한다. 하나의 양자 비트는 수식 (1)과 (2)에서의 한 쌍의 복소수 (α,β)로 정의할 수 있으며, 다음과 같이 표현할 수 있다.

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

그리고 m-양자 비트의 표현은 다음과 같이 정의한다.

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

이때 $|\alpha_i|^2+|\beta_i|^2 = 1,\ i=1,2,\cdots,m$의 조건을 만족한다. 이 표현법은 여러 상태의 중첩을 표현할 수 있는 장점을 갖는다. 예를 들어 다음과 같은 세 쌍의 크기를 갖는 3-양자 비트 시스템이 있다고 가정하자.

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

이 시스템의 상태는 다음과 같이 표현될 수 있다.

$$\begin{aligned} &\frac{1}{4}|000>+\frac{\sqrt{3}}{4}|001>-\frac{1}{4}|010>-\frac{\sqrt{3}}{4}|011> \\ &+\frac{1}{4}|100>+\frac{\sqrt{3}}{4}|101>-\frac{1}{4}|110>-\frac{\sqrt{3}}{4}|111> \end{aligned} \tag{5}$$

위의 결과는 각 상태 $|000>,|001>,|010>,|011>,|100>,|101>,|110>,|111>$을 나타내는 확률이 각각 $\frac{1}{16},\frac{3}{16},\frac{1}{16},\frac{3}{16},\frac{1}{16},\frac{3}{16},\frac{1}{16},\frac{3}{16}$이 된다는 뜻이다. 이것은 (4)와 같은 3-양자 비트 시스템이 동시에 8 가지의 상태를 나타낸다는 것을 의미한다.

양자 비트 표현법을 갖는 진화 알고리즘은 위와 같이 여러 상태의 중첩을 표현할 수 있기 때문에 기존의 접근 방법보다 더 좋은 다양성(diversity)의 특성을 갖는다. 양자 비트 표현법에서는 (4)와 같은 단 하나의 양자 개체가 8 가지 상태를 표현하기에 충분하지만, 기존의 표현법에서는 적어도 8 가지의 염색체 (000), (001), (010), (011), (100), (101), (110), (111)를 필요로 하게 된다. 수렴성(convergence) 또한 양자 비트 표현법에서 얻을 수 있다. $|\alpha_i|^2$ 또는 $|\beta_i|^2$의 값이 '1' 또는 '0'으로 접근함에 따라, 그 양자 개체는 하나의 상태로 수렴해 가게 되며, 점차로 다양성(diversity)은 사라지게 된다. 즉, 양자 비트 표현법은 진화 알고리즘에서의 탐사(exploration)와 개척(exploitation) 두 가지 특성의 균형을 자연스럽게 이룰 수 있다.

2. 양자 진화 알고리즘(QEA)의 구조

제안하는 QEA 알고리즘은 양자 연산의 양자 비트, 상태의 중첩 등과 같은 개념과 원리를 바탕으로 한다. QEA는 개체를 나타내기 위해서 양자 비트 표현법을 사용한다. 다음은 QEA 구조를 수도 코드로 표현한 것이다.

```
procedure QEA
begin
    t ← 0
    Q(t) 초기화
    Q(t) 상태 관측에 의한 P(t) 생성
    P(t) 평가
    P(t) 중 최상의 해 B(t)에 저장
    while (not 종료 조건) do
    begin
        t ← t+1
        Q(t−1) 상태 관측에 의한 P(t) 생성
        P(t) 평가
        양자 연산자 U(t) 이용한 Q(t) 갱신
        P(t) 중 개체별 최상해 B(t)에 저장
        B(t) 중 최상해 b에 저장
```

if 공진 주기

then Q(t) 공진 유도 (**b**를 $B(t)$로 복사)

end

end

QEA는 유전자 알고리즘과 유사한 확률적인 알고리즘이다. QEA는 다음과 같은 개체군을 유지한다: $Q(t) = \{ \mathbf{q}_1^t, \mathbf{q}_2^t, \cdots, \mathbf{q}_n^t \}$, 이때 t는 세대(generation), n은 총 개체수(population size)를 의미한다. 또한 $\mathbf{q}_j^t$는 양자 개체로서 다음과 같이 정의될 수 있다.

$$\mathbf{q}_j^t = \left[\begin{array}{c|c|c|c} \alpha_1^t & \alpha_2^t & \cdots & \alpha_m^t \\ \beta_1^t & \beta_2^t & \cdots & \beta_m^t \end{array} \right] \tag{6}$$

이때 m은 양자 비트의 개수, 즉, 양자 비트의 스트링 길이를 말하고, $j=1,2,\cdots,n$ 이다.

QEA의 양자 개체는 확률적 표현으로, 기존의 진화 알고리즘의 결정적인(deterministic) 표현과는 큰 차이가 있다. 이는 해(solution)와 개체간에 일대일 대응이 되어야 한다는 고정관념에서 벗어나, 해와 개체의 분리를 이룬 것이다. QEA의 양자 개체는 세대가 지남에 따라 해(solution)의 내력 정보를 포함하게 된다.

'$Q(t)$ 초기화' 단계에서는, $Q(t)$내의 모든 벡터 $\mathbf{q}_j^t$ 의 α_i^t와 β_i^t 값을 $\frac{1}{\sqrt{2}}$로 초기화시킨다. 이것은 하나의 양자 개체 $\mathbf{q}_j^t|_{t=0}$ 가 모든 가능한 상태를 같은 확률로 선형 중첩의 상태로 표현한다는 것을 의미한다:

$$|\Psi_{\mathbf{q}_j^0}\rangle = \sum_{k=1}^{2^m} \frac{1}{\sqrt{2^m}} |S_k\rangle$$

이때 S_k는 이진 스트링 $(x_1 x_2 \cdots x_m)$으로 표현되는 k번째 상태를 말한다. x_i는 0 또는 1의 값을 갖는다. 그 다음 단계에서는 이진 해의 집합 $P(t)$를 $Q(t)$를 관측함으로써 생성한다. 세대 t 에서의 이진 해 집합은 $P(t) = \{ P_1^t, P_2^t, \cdots, P_n^t \}$ 이며, $P_j^t = \{ \mathbf{x}_{j1}^t, \mathbf{x}_{j2}^t, \cdots, \mathbf{x}_{jp}^t \}$로 표현한다. 이때 p는 하나의 양자 개체에서 생성하는 이진 해의 개수를 나타낸다. 하나의 이진 해 $\mathbf{x}_{jl}^t$ 은 길이가 m인 이진 스트링이며 ($l=1,2,\cdots,p$), i번째 비트(bit)는 양자 비트 $\mathbf{q}_j^t$의 i번째 확률인 $|\alpha_i^t|^2$과 $|\beta_i^t|^2$의 값을 이용하여 결정한다. 또한 각 이진 해 $\mathbf{x}_{jl}^t$는 적합도(fitness)의 측정을 위해 평가된다. 그리고 나서 각 개체에 해당하는 가장 좋은 해들은 이진 해 $P(t)$ 중에서 선택되어 $B(t)$에

저장된다. 이때 $B(t) = \{ \mathbf{b}_1^t, \mathbf{b}_2^t, \cdots, \mathbf{b}_n^t \}$ 이고, $\mathbf{b}_j^t$ 는 P_j^t 에서 선택된 가장 좋은 이진 해를 말한다. 그림 1은 QEA의 양자 개체와 각 변수간의 관계를 나타낸다.

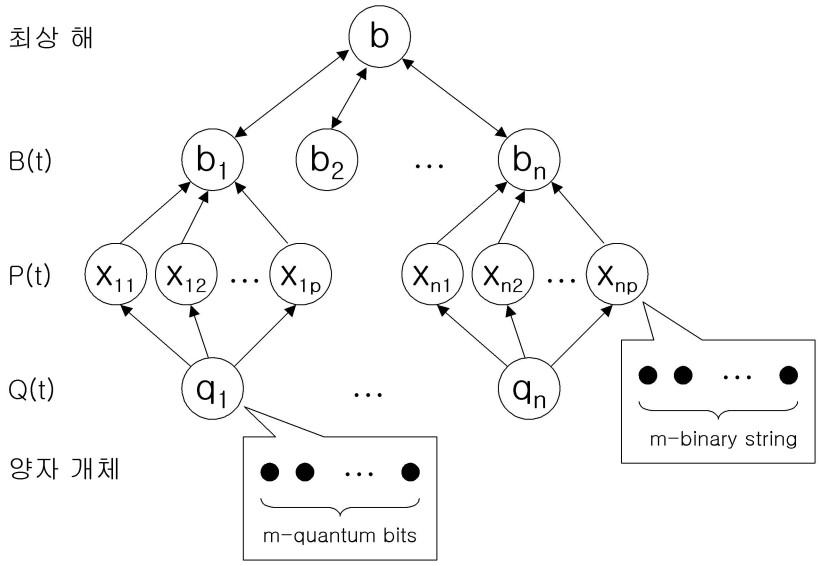


그림 1. 양자 개체와 각 변수간의 관계: 양자 개체군 Q(t), 이진해 집합 P(t), 최상해 집합 B(t), 최상해 b 간의 관계.

while 루프 내에서는, 이진 해의 집합 $P(t)$는 $Q(t-1)$의 관측 행위에 의해 생성되고, 각 이진 해는 적합도(fitness)를 위해 평가된다. 다음 과정인 '$Q(t)$ 갱신'에서는 양자 개체 집합 $Q(t)$가 적합한 양자 연산자[1] $U(t)$의 적용에 의해 갱신된다. 이때 양자 연산자는 이진 해 집합 $P(t)$와 저장된 베스트 솔루션 $B(t)$의 정보를 이용하여 형성할 수 있다. 적합한 양자 연산자는 실제적인 문제에 따라 결정될 수 있다. QEA에서 가장 기본이 되는 양자 연산자는 다음과 같은 회전 연산자이다.

$$U(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{7}$$

이 단계는 양자 개체가 보다 적합한 상태로 수렴하도록 만들어 준다. 다음 단계에서 $P(t)$ 중에 가장 좋은 해들이 $B(t)$에 저장되고, 만약 $B(t)$ 중에서 현재 저장되어 있는 최상해 $\mathbf{b}$보다 더 적합할 경우에는 최상해가 교체된다. 다음 단계에서 만약 공진 주기에 해당하면 최상해 $\mathbf{b}$와 각 개체에서의 최상해 $B(t)$간의 공진을 유도한다. 이때 공진 주기

1) 양자 연산자는 양자 게이트(quantum gate)를 이용한다. 양자 게이트는 가역적 게이트(reversible gate)이고, 양자 비트 기반 상태에 적용되는 다음과 같은 관계를 만족하는 유니터리 오퍼레이터(unitory operator)로 표현될 수 있다: $U^{\dagger} U = UU^{\dagger}$, 이때 $U^{\dagger}$ 는 U의 hermitian adjoint를 의미한다. 양자 게이트에는 NOT 게이트, Controlled NOT 게이트, Rotation 게이트, Hadamard 게이트 등이 있다.

는 문제에 따라 임의로 설정할 수 있으며, 공진이란 양자 개체가 현재의 최상해 **b**의 상태를 표현하도록 양자 비트들의 확률이 변하는 과정으로 정의한다. 부분 공진 유도와 전체 공진 유도로 분류할 수 있다. 전체 공진 유도는 최상해 **b**값을 $B(t)$의 내의 모든 개체 최상해에 복사하고, 부분 공진 유도는 최상해 **b**를 $B(t)$내의 몇 개의 개체 최상해에만 복사하는 것으로 구현할 수 있다. 이진해의 집합 $P(t)$는 루프의 마지막에 버려진다.

3. 양자 진화 알고리즘(QEA)의 특징

앞 절에서 제안된 양자 진화 알고리즘(QEA)의 특징을 알아보기 위해 간단한 배낭 문제에 적용해 본다. 배낭 문제는 III장에서 설명한다. 총 10개의 아이템이 있을 경우, 아이템을 선택할 수 있는 경우의 수는 2^{10}, 즉 1024가지가 된다. 그림 2는 1024가지 경우에 대한 각각의 이득 값을 나타낸다. 이때 배낭의 용량을 만족하는 최적의 이득 값은 62.192938 이고, 127번째 경우이다.

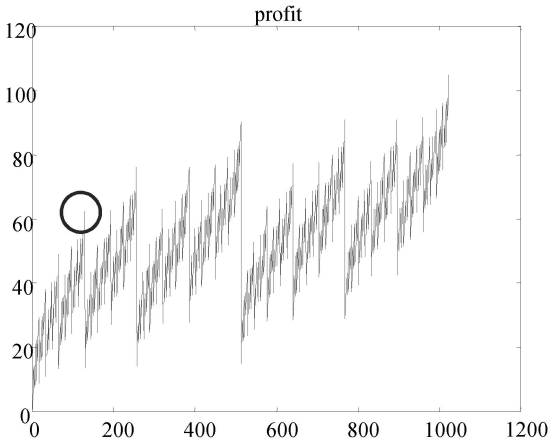


그림 2. 아이템 10개를 갖는 배낭 문제에서 1024 가지의 모든 경우에 대한 이득 값.
(배낭의 제한 조건에 의해 O 표시된 부분이 최적해이다.)

QEA의 특성을 파악하기 위해, 양자 개체 1개, 즉 population 크기를 1로 설정하고, 양자 개체에서 생성하는 이진 해의 개수도 1개로 설정하였다. 그림 3부터 그림 10까지는 QEA 연산 과정 중 양자 개체의 확률을 도시한 것이다. 각 그림은 10, 20, 30, 40, 50, 100, 200, 300 세대에서의 결과이다. 그림 3과 그림 4에서 0.001 근처의 직선은 비교를 위해 초기의 양자 개체 확률을 함께 표시한 것이다. 즉, 초기의 양자 개체는 모든 경우의 상태를 동일한 확률로 포함하고 있다. 이는 초기에 무작위 검색으로 시작한다는 것을 의미한다.

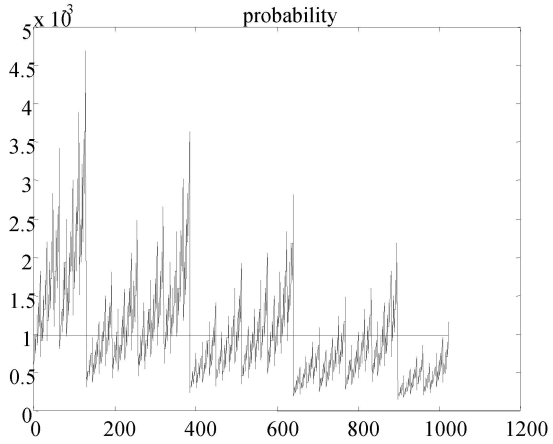


그림 3. 양자 개체의 확률(10 generations), 0.001 근처의 직선은 초기의 양자 개체 확률을 나타낸다.

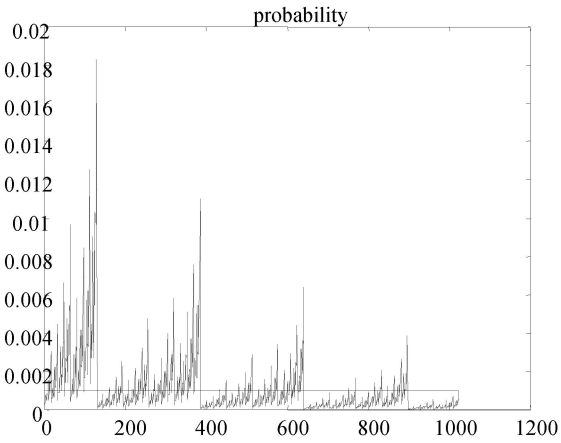


그림 4. 양자 개체의 확률(20 generations), 0.001 근처의 직선은 초기의 양자 개체 확률을 나타낸다.

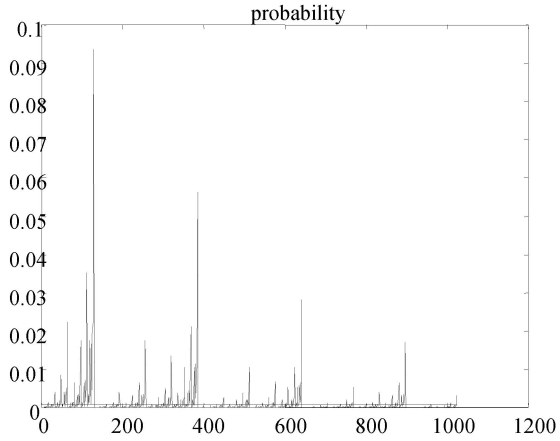


그림 5. 양자 개체의 확률(30 generations)

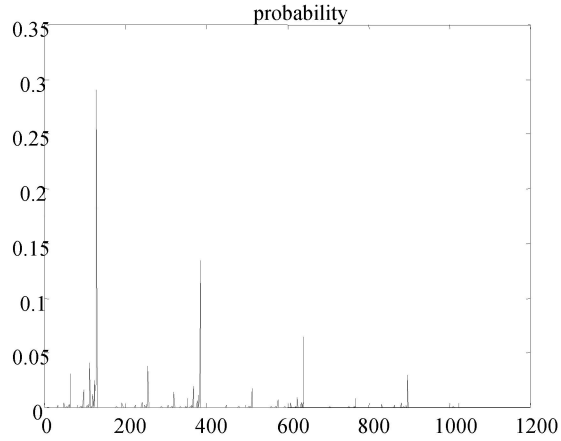


그림 6. 양자 개체의 확률(40 generations)

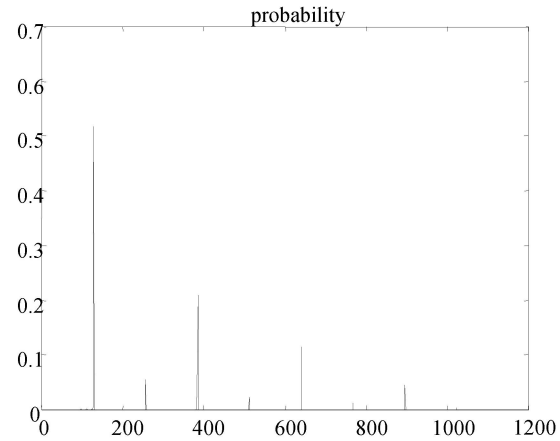


그림 7. 양자 개체의 확률(50 generations)

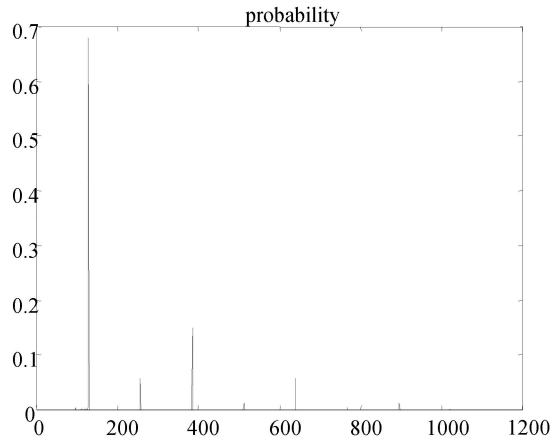


그림 8. 양자 개체의 확률(100 generations)

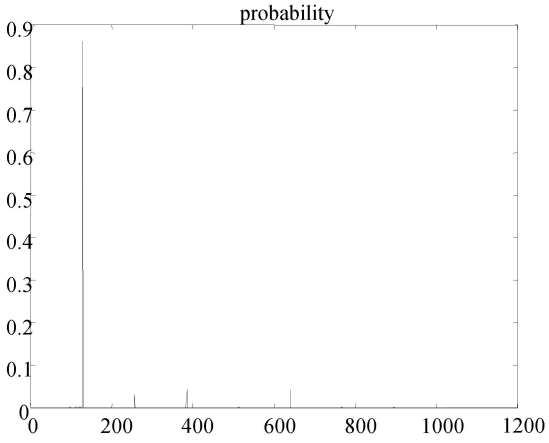


그림 9. 양자 개체의 확률(200 generations)

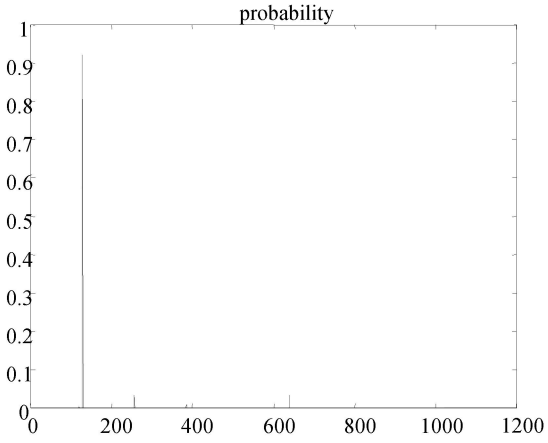
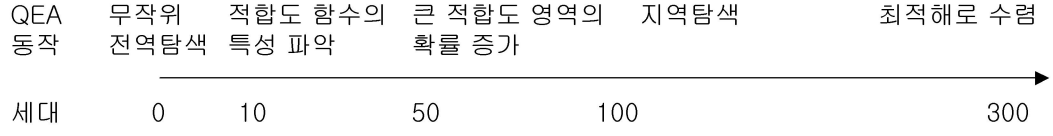


그림 10. 양자 개체의 확률(300 generations)

세대 10의 결과인 그림 3을 보면 QEA의 흥미로운 특징을 알 수 있다. 양자 개체의 개수가 1개임에도 불구하고, 양자 개체의 확률이 그림 2의 이득 특성을 그대로 표현하고 있다. (단, 그림 2는 점차 값이 증가하는데 그림 3에서는 점차 떨어지는 이유는 배낭의 용량 제한을 초과했기 때문이다.) 즉, 단 1개의 양자 개체로 이득 함수의 정보를 모두 포함하고 있는 것이다. 세대 20에서는 이득 함수 특성을 그대로 유지한 상태에서 전체적으로 확률이 커졌음을 확인할 수 있다. 세대 30에서는 이득이 큰 부분의 확률이 크게 증가한 것을 알 수 있다. 세대 40에서는 이득이 큰 부분 이외의 다른 부분은 거의 확률의 증가가 눈에 띄지 않는다. 세대 50까지는 이득이 큰 부분의 확률이 계속 증가했지만, 세대 100에서 보면 최적해 위치를 제외한 다른 부분의 피크 값이 줄어들었음을 알 수 있다. 세대 200에서도 같은 특성을 보이고 있고, 세대 300에서는 최적해 위치의 확률이 90%를 넘고, 나머지 부분의 확률은 거의 0에 가까워 진 것을 확인할 수 있다. 즉, 세대 300에서는 양자 개체가 최적해로 거의 수렴한 것이다.

위의 결과에서 나타난 QEA의 동작은 다음과 같이 정리할 수 있다. 초기 무작위 전역탐색으로 시작하여, 전체 영역의 적합도 분포 특성을 파악하고, 적합도 큰 영역에 대한 확률 증가 후, 점차적으로 지역탐색을 하게 되며, 결국은 최적해로 수렴하게 된다.



즉, QEA는 초기 전역탐색에서 세대가 증가하여 우수한 해가 찾아짐에 따라 자동적으로 지역탐색으로 동작이 바뀌게 되며, 결국은 최적해 근처의 값으로 수렴을 이루는 특징을

갖는다. 또한 양자 개체가 1개임에도 불구하고, 전체 함수의 특성을 모두 파악하여 최적해를 찾아갈 수 있는 것은 확률적 표현에 의해 진화 과정의 내력 정보가 모두 양자 개체에 흡수되기 때문이다. QEA의 장점들은 다음과 같이 정리할 수 있다.

- QEA의 구조적 특성에 의해, 전역탐색과 지역탐색의 균형이 자동적으로 이루어진다. 즉, 전역탐색과 지역탐색을 위한 별도의 휴리스틱 등이 필요치 않다.
- 확률적 표현에 의해 진화 과정의 내력 정보가 양자 개체에 흡수되기 때문에, 우성의 스키마를 정확히 알 수 있다.
- 하나의 양자 개체는 여러 상태 중첩을 표현하므로, 적은 개체수로도 우수한 해를 얻을 수 있다.
- 적은 개체수를 이용하므로 짧은 시간 내에 우수한 해를 얻을 수 있다.
- 최적에 가까운 해가 찾아졌을 경우 양자 개체가 자동적으로 수렴하기 때문에, 알고리즘의 종료 조건을 쉽게 설정할 수 있다. 종료 조건은 다음과 같이 설정이 가능하다.

$$\text{종료 조건: } Prob(\mathbf{b}) > \gamma$$

예를 들어 $\gamma=0.9$일 경우, 위의 문제에서는 300 세대 이전에 알고리즘을 종료하게 된다. 최상해의 확률 $Prob(\mathbf{b})$ 값은 양자 개체의 확률을 이용하여 쉽게 구할 수 있다.

III. 배낭 문제(knapsack problem)에의 적용

QEA의 성능을 알아보기 위해 조합 최적화 문제의 일종인 배낭(knapsack) 문제를 도입한다. 배낭 문제는 많은 아이템 중에서 가장 이득이 되는 아이템의 조합을 고르는 문제로, 고를 수 있는 총 아이템의 용량이 제한되어 있다. 다시 정리하면, m개의 아이템과 하나의 주머니(knapsack)가 있을 때, 다음의 이득 $f(\mathbf{x})$을 최대로 하는 아이템들을 선택하는 문제이다:

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

주머니(knapsack)의 용량은 C이며, 다음의 조건을 만족해야 한다:

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

이때 $\mathbf{x} = (x_1 \cdots x_m)$이며, x_i는 0 또는 1의 값을 갖는다. 또한 p_i는 아이템 i의 이득이며, w_i는 아이템 i의 무게를 나타낸다.

본 장에서는, 배낭 문제를 위한 몇 개의 전통적인 유전자 알고리즘(GA: genetic

algorithm)에 대한 설명과 배낭 문제를 위한 자세한 QEA 알고리즘에 대해 언급한다.

1. 전통적인 GA 방법들

여기서는 세 가지 전통적인 알고리즘을 기술하고 테스트한다. 페널티(penalty) 함수를 기반으로 하는 알고리즘, 리페어(repair) 방법을 기반으로 하는 알고리즘, 그리고 디코더(decoders)를 기반으로 하는 알고리즘에 대해서 다룬다[1].

페널티 함수를 기반으로 하는 모든 알고리즘에 있어서, 길이가 m인 이진 스트링은 하나의 염색체 $\mathbf{x}$를 나타낸다. 각 스트링의 이득 $f(\mathbf{x})$는 다음과 같이 결정된다:

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

이때 $Pen(\mathbf{x})$는 페널티 함수이다. 페널티 함수를 결정하기 위한 많은 가능한 전략들이 있다[12,13]. 여기서는 다음과 같은 2 가지 종류, 즉 logarithmic 페널티, linear 페널티의 페널티 함수를 고려한다.

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

이때 ρ는 $\max_{i=1\cdots m}\{p_i/w_i\}$ 이다.

리페어(repair) 방법을 기반으로 하는 알고리즘에서는 각 스트링의 이득 $f(\mathbf{x})$이 다음과 같이 결정된다:

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

이때 $\mathbf{x}'$는 원래의 벡터 $\mathbf{x}$의 고쳐진 벡터이다. 본 실험에서는 원래의 염색체가 5%의 확률을 가지고 수정된다. 리페어 과정은 다음과 같이 나타낼 수 있다.

procedure repair ($\mathbf{x}$)

begin

배낭 용량 넘침 여부 ← *false*

$\mathbf{x}' \leftarrow \mathbf{x}$

if $\sum_{i=1}^{m} w_i x_i' > C$

then 배낭 용량 넘침 여부 ← *true*

while (배낭 용량 넘침 여부) **do**

begin

배낭에서 i 번째 아이템 선택

$x_i' \leftarrow 0$

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

then 배낭 용량 넘침 여부 $\leftarrow$ *false*

end

end

여기서 고려된 두 가지의 리페어 알고리즘은 단지 제거할 아이템을 고르는 선택 과정에서만 차이가 있다:

Rep_1(random repair): 선택 과정은 주머니에서 임의의 아이템을 선택한다.

Rep_2(greedy repair): 주머니 안의 모든 아이템은 무게 당 이득의 비율이 작아지는 순서로 정렬되고, 선택 과정에서는 제거할 아이템으로 항상 마지막 아이템을 선택한다.

배낭 문제를 위해 가능한 디코더(decoder)는 정수 표현법을 기반으로 한다. 각 염색체는 m개의 정수의 벡터이며, 벡터의 i번째 요소는 1과 $m-i+1$ 사이의 정수이다. 이와 같이 차례를 나타내는 표현법은 아이템들의 리스트 L을 참조한다. 즉, 현재의 리스트로부터 적당한 순서에 해당하는 아이템을 선택하는 방법으로 벡터가 해석된다. 다음은 디코딩(decoding) 알고리즘을 나타낸다:

procedure decoding (x)

begin

아이템 리스트 L 생성

$i \leftarrow 1$

무게합 $\leftarrow 0$

이득합 $\leftarrow 0$

while $i \leq m$ **do**

begin

$j \leftarrow x_i$

리스트 L에서 j 번째 아이템 제거

if 무게합 $+ w_j \leq C$ **then**

begin

무게합 ← 무게합 + w_j

이득합 ← 이득합 + p_j

end

$i \leftarrow i+1$

end

end

Dec_1(random decoding): 리스트 생성 과정에서 랜덤한 입력 파일의 아이템 순서를 그대로 갖는 리스트 L을 생성한다.

2. 배낭 문제를 위한 QEA

배낭 문제를 위한 QEA 알고리즘은 앞에서 제안한 기본 QEA 구조를 기반으로 하고, 리페어 알고리즘을 포함한다. 구체적인 알고리즘은 다음과 같다:

procedure QEA

begin

$t \leftarrow 0$

$Q(t)$ 초기화

$Q(t)$ 상태 관측에 의한 $P(t)$ 생성

$P(t)$ 고침 (repair $P(t)$)

$P(t)$ 평가

$P(t)$ 중 최상의 해 $B(t)$에 저장

while (not 종료 조건) **do**

begin

$t \leftarrow t+1$

$Q(t-1)$ 상태 관측에 의한 $P(t)$ 생성

$P(t)$ 고침 (repair $P(t)$)

$P(t)$ 평가

양자 연산자 $U(t)$ 이용한 $Q(t)$ 갱신

$P(t)$ 중 개체별 최상해 $B(t)$에 저장

$B(t)$ 중 최상해 **b**에 저장

if 공진 주기

then Q(t) 공진 유도 (**b**를 $B(t)$로 복사)

end

end

(6)과 같이 표현되는 길이가 m인 양자 비트 스트링은 선형 중첩된 해를 표현한다. i번째 아이템은 $|\beta_i|^2$ 또는 $(1-|\alpha_i|^2)$의 확률을 가지고 선택될 수 있다. 양자 비트 스트링의 길이는 아이템의 개수와 일치한다. 길이가 m인 이진 스트링은 양자 비트 스트링으로부터 생성된다. 이진 스트링의 각각의 비트(bit)에 대해서는, 0과 1 사이의 랜덤 넘버 r을 발생시켜 $r > |\alpha_i|^2$의 조건을 만족하면 이진 스트링의 i번째 비트를 1로 설정한다. 이진 스트링 **x**는 문제의 실질적인 해를 표현하며, 만약 $x_i = 1$ 이면 i번째 아이템이 선택된 것을 의미한다. 이진 스트링 **x**는 다음과 같은 과정으로 생성된다:

procedure make (x)

begin

$i \leftarrow 0$

while ($i < m$) **do**

begin

$i \leftarrow i+1$

if $random[0,1) < |\alpha_i|^2$

then $x_i \leftarrow 0$

else $x_i \leftarrow 1$

end

end

배낭 문제를 위한 QEA에서 사용된 리페어 알고리즘은 다음과 같이 구현되었다:

procedure repair (x)

begin

배낭 용량 넘침 여부 ← *false*

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

then 배낭 용량 넘침 여부 ← *true*

while (배낭 용량 넘침 여부) **do**

begin

배낭에서 i 번째 아이템 선택

$x_i \leftarrow 0$

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

then 배낭 용량 넘침 여부 ← *false*

end

while (not 배낭 용량 넘침 여부) **do**

begin

배낭에서 j 번째 아이템 선택

$x_j \leftarrow 1$

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

then 배낭 용량 넘침 여부 ← *true*

end

$x_j \leftarrow 0$

end

이진 해 **x**의 이득은 $\sum_{i=1}^{m} p_i x_i$ 으로 평가되며, 양자 개체 **q**의 갱신 이후에 가장 좋은 솔루션 **b**를 찾는데 이용된다. 배낭 문제를 위한 본 알고리즘에서는 양자 개체 **q**를 갱신하기 위해 (7)과 같은 회전 연산자를 사용한다. i번째 양자 비트 값 (α_i, β_i)는 다음과 같이 바뀌게 된다:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \quad (8)$$

본 배낭 문제에서는 θ_i를 $s(\alpha_i\beta_i)\Delta\theta_i$ 로 결정한다. 실험에서 사용된 파라메터 값은 표 1과 같다. 만약 조건 $f(\mathbf{x}) \geq f(\mathbf{b})$을 만족하고, x_i와 b_i 값이 각각 1과 0일 경우, 우리는

$\Delta\theta_i$를 0.025π로, $s(\alpha_i\beta_i)$는 상태 |1>의 확률을 증가시키는 방향으로 $\alpha_i\beta_i$의 조건에 따라 +1, -1, 또는 0으로 설정할 수 있다. $\Delta\theta_i$의 값은 수렴 속도에 영향을 미친다. 하지만 만약 값이 너무 크게 되면, 그 해는 발산하거나 지역 해(local optimum)에 머무를 수 있게 (premature convergence) 된다. 부호 $s(\alpha_i\beta_i)$는 전역 최적해를 향한 수렴 방향을 결정하게 된다. 표 1은 단지 θ_i 결정 방법의 한 가지 예를 보인 것이다.

표 1. θ_i 참조표: $f(\cdot)$는 이득 함수, $s(\alpha_i\beta_i)$는 θ_i의 부호, b_i와 x_i는 각각 베스트 솔루션 **b**와 이진 솔루션 **x**의 i번째 비트를 나타낸다.

x_i	b_i	$f(\mathbf{x}) \geq f(\mathbf{b})$	$\Delta\theta_i$	$s(\alpha_i\beta_i)$			
				$\alpha_i\beta_i>0$	$\alpha_i\beta_i<0$	$\alpha_i=0$	$\beta_i=0$
0	0	false	0	0	0	0	0
0	0	true	0	0	0	0	0
0	1	false	0	0	0	0	0
0	1	true	0.05 π	-1	+1	$\pm$1	0
1	0	false	0.01 π	-1	+1	$\pm$1	0
1	0	true	0.025 π	+1	-1	0	$\pm$1
1	1	false	0.005 π	+1	-1	0	$\pm$1
1	1	true	0.025 π	+1	-1	0	$\pm$1

다음은 갱신(update) 과정을 나타낸다:

procedure update (q)

begin

$i \leftarrow 0$

while ($i < m$) **do**

begin

$i \leftarrow i+1$

θ_i 값 결정

(α_i', β_i') 연산:

$$[\alpha_i' \ \beta_i']^T = U(\theta)[\alpha_i \ \beta_i]^T$$

end

$\mathbf{q} \leftarrow \mathbf{q}'$

end

갱신(update) 과정은 적합한 양자 게이트들을 이용하여 다양한 방법으로 구현이 가능하다. 이것은 주어진 문제에 달려있다.

IV. 실험 결과

모든 실험은 다음과 같은 강한 연관성을 갖는 데이터 집합을 이용한다:

$$w_i = uniformly\ random[1, 10)$$
$$p_i = w_i + 5$$

또한 주머니의 용량은 다음과 같은 평균 용량을 사용한다:

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

모든 사용하는 데이터 파일은 정렬되어 있지 않다. 여섯 가지 전통적인 유전자 알고리즘(CGAs)의 population size는 모두 100이다. 교배와 돌연변이의 확률은 각각 0.05와 0.01로 고정시켰다[1]. QEA1과 QEA2의 population size, 즉 양자 개체 수는 10이다. QEA1과 QEA2의 차이점은 공진 유도 단계의 적용 여부이다. 공진 유도가 미치는 영향을 확인하기 위한 것이다. QEA2에서는 양자 개체 10개를 2개씩 묶어 5개의 그룹으로 나누고, 각 그룹들간에는 공진 유도가 발생한다. 공진 주기는 50세대로 설정하였다. 또한 전체 공진 유도를 적용하였다. 알고리즘의 성능을 평가하기 위해서 1000 세대 내에서의 가장 좋은 솔루션을 기록하고, 총 30번의 테스트를 수행했다. 또한 한 번 수행하는 걸리는 평균 시간을 측정하였다. 모든 실험은 펜티엄 III 800MHz 프로세서를 사용하였으며, 프로그래밍 툴로 Visual C++ 6.0을 이용하였다. (본 실험은 문제의 정확한 최적해를 구하기보다는, QEA의 기본 구조만을 적용했을 때 얼만큼의 성능을 보이는지를 관찰하기 위한 것이다.)

표 2. 배낭 문제의 실험 결과: 아이템의 개수는 100, 250, 500. 최대 세대수는 1000, 테스트 회수는 30번이고, P2+R1은 Pen2와 Rep1 방법을 동시에 사용한 방법을 의미한다. b., m., w.은 각각 best, mean, worst를 의미하고, t(sec/run)은 한 번 테스트에 소요되는 평균 시간을 나타낸다. QEA1과 QEA2의 차이는 공진 유도 주기의 적용 여부이다.

			CGAs						QEAs	
			Pen1	Pen2	Rep1	Rep2	Dec1	P2+R1	QEA1	QEA2
100	Profits	B.	562.1	597.6	558.9	560.5	512.3	592.7	612.5	612.7
		M.	549.2	587.7	547.2	545.3	502.1	586.8	604.1	607.3
		W.	540.0	572.6	537.1	536.8	493.5	577.6	592.7	601.7
	t(sec/run)		1.537	1.544	1.293	1.302	12.97	1.556	0.408	0.438
250	Profits	B.	1377.7	1455.0	1383.1	1352.7	1179.7	1454.0	1497.3	1505.2
		M.	1341.3	1439.0	1343.1	1337.4	1159.4	1441.7	1473.2	1490.5
		W.	1321.1	1415.2	1319.5	1322.5	1140.7	1430.1	1433.9	1473.4
	t(sec/run)		3.642	3.739	3.040	3.116	69.29	3.754	1.566	1.793
500	Profits	B.	2712.4	2839.6	2706.8	2686.3	2255.1	2839.1	2903.2	2909.7
		M.	2668.4	2804.5	2661.0	2657.4	2220.9	2805.1	2858.7	2880.5
		W.	2642.8	2766.3	2626.6	2628.1	2195.7	2781.0	2821.1	2854.5
	t(sec/run)		9.211	9.274	7.913	8.142	270.8	9.138	4.429	5.981

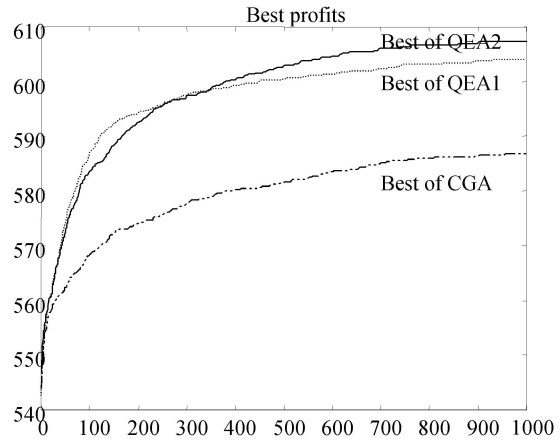
표 2의 결과를 보면, QEA는 모든 다른 CGA에 비해서 우수한 것을 확인할 수 있다. CGA중에서 선형 페널티 함수와 랜덤 리페어 알고리즘을 동시에 적용한 알고리즘이 가장 우수한 결과를 보였다. 하지만 이 결과는 QEA2 뿐만 아니라 QEA1 보다도 나쁜 결과를 보이고 있다. 이것은 QEA 알고리즘이 population size가 작음에도 불구하고 더 좋은 결과를 보인다는 것을 입증해 준다. 이 실험 결과로 판단해 보면, QEA는 CGA에 비해서 더 짧은 시간 안에 더 우수한 해을 찾을 수 있다. 아이템 100, 250, 500개 모든 경우에 대해 실험 결과는 QEA의 우수성을 입증하고 있다.

그림 11은 QEA1, QEA2, CGA 각각에 대해 30번의 테스트에 걸친 최상 이득의 평균과 모든 개체의 평균 이득의 평균의 추이를 100, 250, 500 아이템에 각각에 대해 보여준다. 이때 CGA는 P2+R1 알고리즘을 의미한다. 수렴 속도 면에서나 최종 결과 면에서 모두 CGA에 비해 QEA가 우수한 성능을 나타내고 있음을 명백히 보여준다. 특히 QEA1과 QEA2의 결과는 흥미롭다. QEA1과 QEA2의 population 크기는 동일하고, 공진 유도 주기의 적용 여부만 다른 경우이다. 초반에는 QEA1이 QEA2에 비해서 더 빠른 속도로 증가하지만, 대략 300 세대에서부터 QEA2가 QEA1을 추월하게 된다. 이는 QEA2의 경우 그룹별로 독립된 진화과정을 거치기 때문에, 탐색에 있어서 다양성을 제공해주는 역할을 하

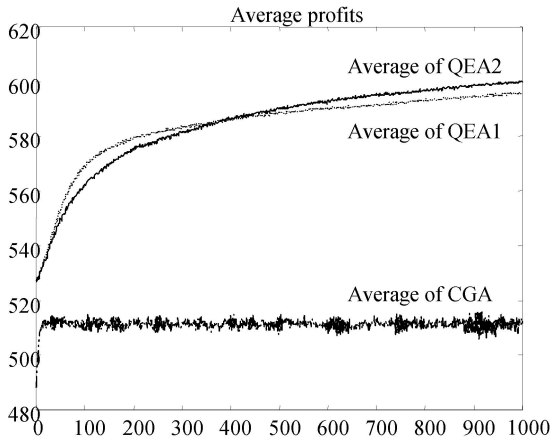
기 때문이다. 또한 공진 유도 주기마다 가장 우수한 해를 모든 그룹에 전달하기 때문에, 국부해에 빠져 있던 상태를 벗어날 수 있는 기회를 제공한다. QEA2는 QEA1보다 개체의 다양성을 제공하기 때문에, 세대가 지남에 따라 더 좋은 결과를 보이는 것이다. 1000 세대에서, QEA의 최종 이득 값은 CGA에 비해 훨씬 더 크다. 수렴성의 경향은 population의 평균 이득(average profits) 결과 그림에서 확실히 나타난다. 시작부분에서는 세 가지 알고리즘이 모두 수렴 속도가 크다. 하지만 CGA는 바로 조기 수렴(premature convergence)으로 인해 거의 상수에 가까운 이득을 유지하게 된다. 반면에 QEA는 거의 일정한 수렴 속도를 가지고 최적해 근처를 향해 증가하는 것을 확인할 수 있다. QEA는 1000세대가 될 때까지 CGA에서 흔히 발생하는 premature convergence를 보이지 않는다. 본 실험 결과는 QEA의 효율성 및 응용성을 입증하며, 특히 그림 11은 QEA의 우수한 전역 탐색 능력과 뛰어난 수렴성을 보여준다.

V. 결론

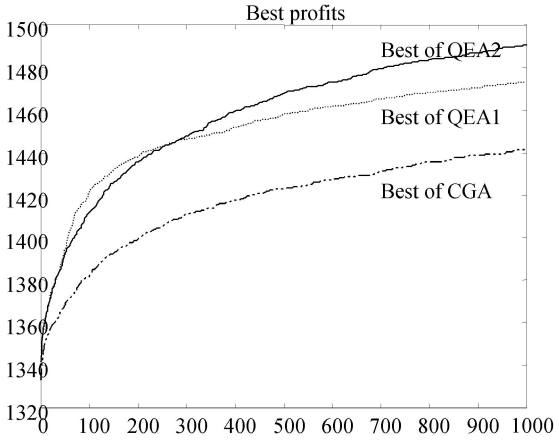
본 논문은 새로운 진화 알고리즘인 양자 진화 알고리즘(QEA: quantum-inspired evolutionary algorithm)을 제안했다. QEA는 양자 비트, 상태의 중첩 등의 양자 연산의 원리와 개념을 바탕으로 한다. QEA의 양자 개체는 여러 상태의 선형 중첩을 표현할 수 있고, 많은 개체를 포함할 필요가 없다. QEA는 양자 비트 표현법에 의한 다양성(diversity)에 의해 뛰어난 전역 탐색 능력과, 진화 과정의 내력을 이용한 빠른 수렴성을 갖는다. 짧은 시간 내에 CGA보다 더 좋은 솔루션에 도달할 수 있다. 또한 QEA는 종료 시점의 명확한 판단 기준을 설정할 수 있다는 장점을 갖는다. QEA의 성능을 평가하기 위해 조합 최적화 문제의 일종인 배낭 문제를 도입했다. 실험은 QEA의 수렴성과 전역 탐색 능력이 CGA에 비해 우수하다는 것을 보여준다.



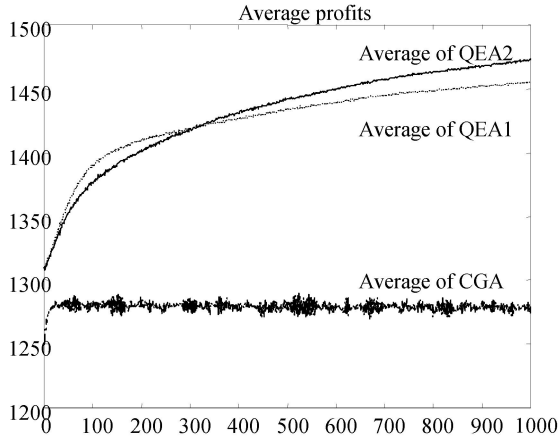
(a) best profits (100 items)



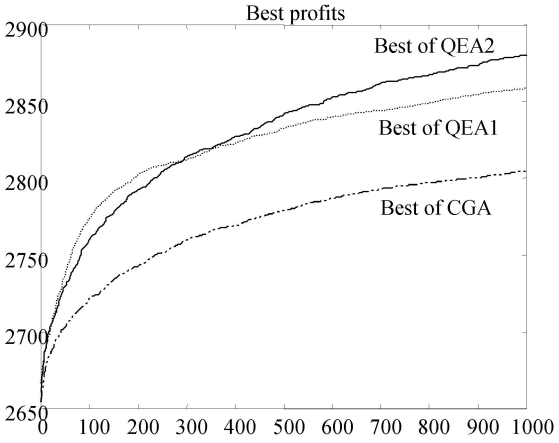
(b) average profits (100 items)



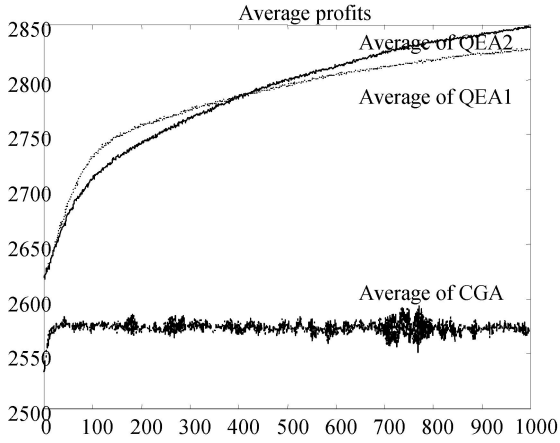
(c) best profits (250 items)



(d) average profits (250 items)



(e) best profits (500 items)



(f) average profits (500 items)

그림 11. 배낭 문제에 대한 CGA와 QEA의 비교: 세로축은 배낭의 이득, 가로축은 세대 수를 의미한다. (a),(c),(e)는 최상 이득의 평균, (b),(d),(f)는 평균 이득의 평균을 나타낸다. 두 경우 모두 30번의 테스트 결과를 평균한 값이다.

참고 문헌

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

[2] 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*, 22, pp. 563-591, 1980.

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

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

[5] P. W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, 1994.

[6] 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.

[7] L. K. Grover, "Quantum Mechanical Searching," in *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 2255-2261, Jul 1999.

[8] 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*, pp. 2239-2246, Jul 1999.

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

[10] T. Hey, "Quantum computing: an introduction," *Computing & Control Engineering Journal*, pp. 105-112, Jun 1999.

[11] R. Hinterding, "Representation, Constraint Satisfaction and the Knapsack Problem," in *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1286-1292, Jul 1999.

[12] J.-H. Kim and H. Myung, "Evolutionary Programming Techniques for Constrained Optimization Problems," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 2, pp. 129-140, Jul 1997.

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