Quantum-Inspired-Evolutionary-Algorithm-Knapsack problem

Author1: Mark Zakharov
Organization: Innopolis University
Group: DS-01
email: ma.zakharov@innopolis.university
Author2: Mahmoud Mousatat
Organization: Innopolis University
Group: DS-01
email: m.mousatat@innopolis.university

Abstract—The quantum-inspired evolutionary algorithm (QEA), which is based on the idea and tenets of quantum computing, such as a quantum bit and superposition of states, is proposed in this study as a revolutionary evolutionary algorithm. The representation of the individual, the evaluation function, and the population dynamics are the same characteristics that distinguish other evolutionary algorithms from QEA. However, QEA employs a Q-bit, which is defined as the lowest unit of information, for the probabilistic representation instead of a binary, numeric, or symbolic representation, and a Q-bit individual is a string of Q-bits. As a variation operator, a Q-gate is included to entice individuals to come up with better solutions.

Experiments are conducted on the well-known combinatorial optimization issue known as the "knapsack problem" to show its efficacy and application. In comparison to the traditional genetic algorithm, the results demonstrate that QEA performs effectively even with a small population without experiencing premature convergence. Index Terms: Knapsack problem, probabilistic representation, quantum computing, evolutionary algorithm.

|Github link for Code Resources: LINK

Introduction

The main component of evolutionary algorithms (EAs) is a stochastic search and optimization technique based on the concepts of organic biological evolution. EAs are robust, global, and may be widely used in contrast to conventional optimization techniques like calculus-based and enumerative procedures, albeit performance is still impacted by these heuristics. Using the survival of the fittest to create ever better approximations to a solution, EAs operate on a population of potential solutions. By selecting people according to their level of fitness in the problem area and reproducing them using variation operators, a new set of approximations is produced at each generation of the EA. Similar to how natural adaptation occurs, this process may result in the evolution of communities of individuals who are more adapted to their environment than the individuals from which they originated. The representation of the individual, the evaluation function that reflects the level of fitness of the individual, and population dynamics such as population size, variation operators, parent selection, reproduction and inheritance, the survival competition method, etc. are what define EAs. Those elements need to be correctly constructed in order to have a fair balance between exploration and exploitation. In particular, the representation and variation operators are examined in this paper to represent the individuals effectively to explore the search space with a small number of individuals (even with just one individual for real-time application) and, respectively, to exploit the global solution in the search space quickly. The suggested evolutionary algorithm incorporates some quantum computing techniques to achieve these goals. The quantum-inspired evolutionary algorithm (QEA), which is proposed in this study, is a revolutionary evolutionary algorithm that is based on the idea and principles of quantum computing, such as a quantum bit and

superposition of states. The representation of the individual, the evaluation function, and the population dynamics both serve to distinguish QEAs from EAs. However, QEA uses a Q-bit as a probabilistic representation rather than a binary, numeric, or symbolic representation, which is referred to as the lowest unit of information. A string of Q-bits serves as the definition of a Q-bit person. The Q-bit individual has the benefit of being able to probabilistically represent a linear superposition of states (binary solutions) in the search space. As a result, compared to other representations, the Q-bit representation has a superior feature of population variability. A Q-gate is also described as a QEA variation operator that directs agents to improve solutions and ultimately to a single state. Since a Q-bit individual represents the linear superposition of all possible states with the same probability, QEA can initially represent a variety of individuals probabilistically. The diversity attribute eventually vanishes as each Q-bit converges to a single state and the probability of each Q-bit approaches either 1 or 0 by the Q-gate. This built-in mechanism allows QEA to manage the equilibrium between exploration and exploitation. Although QEA is founded on the idea of quantum computing, it should be underlined that QEA is not a quantum algorithm but rather a cutting-edge evolutionary algorithm for a classical computer [5], [4].

Related Work

This paper is based on Quantum-inspired evolutionary algorithm for a class of combinatorial Optimization byKuk-Hyun Han and Jong-Hwan Kim, which I took The Algorithm structure and built the code based on it and compared the result with a standard GA for knapsack problems.

QEA

Before describing QEA, the basics of quantum computing are briefly addressed in the following. The smallest unit of information stored in a two-state quantum computer is called a quantum bit, or qubit [6]. A qubit may be in the “1” state, in the “0” state, or in any superposition of the two. The state of a qubit can be represented as

$$\psi> = \alpha 0> + \beta> \qquad (1)$$

Where $\alpha \; and \; \beta$ are the complex integers that represent the associated states' probability amplitudes. The likelihood that a qubit will be found in the "0" state is given by α^2 , and the likelihood that a qubit will be found in the "1" state is given by β^2. Guaranteed by normalizing the state to oneness:

$$\alpha^2 + \beta^2 = 1 \qquad (2)$$

A.Representation

For the probabilistic representation that is based on the idea of qubits, QEA employs a novel representation known as a Q-bit, and a "Q-bit individual," which is defined as a string of Q-bits.

Definition 1: The smallest unit of information in QEA is known as a "Q-bit," and it is defined with a pair of numbers as (α , β). A Q-bit can be in a linear superposition of the states "1," "0," or both.

Definition 2: As a group of Q-bits, a Q-bit individual is defined as:

$$\begin{bmatrix} \alpha_1 \; \alpha_2 \alpha_3 \ldots . \alpha_m \\ \beta_1 \beta_2 \beta_3 \ldots . \beta_m \end{bmatrix} \qquad (3)$$

Where $\alpha i^1 + \beta i^2 = 1$,i = 1,2,3,..m.

The ability to represent a linear superposition of states in a Q-bit representation is a benefit. If there is a three-Q-bit system, for instance, with three pairs of amplitudes, as in:

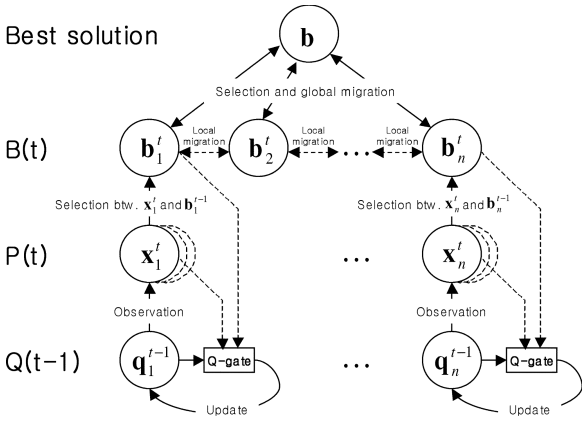
$$\begin{bmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2} \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, \frac{\sqrt{3}}{2} \end{bmatrix} \qquad (4)$$

then the system's states can be depicted as:

$$\frac{1}{4}|000>+\frac{\sqrt{3}}{4}|001>-\frac{1}{4}|010>-\frac{\sqrt{3}}{4}|011>$$
$$+\frac{1}{4}|100>+\frac{\sqrt{3}}{4}|101>-\frac{1}{4}|110>-\frac{\sqrt{3}}{4}|111> \quad (5)$$

Since it can express linear superposition of states probabilistically, evolutionary computing with Q-bit representation has a better feature of population variety than other representations. Eight states can be represented by a single Q-bit individual, such as (4), but in binary representation, at least eight strings are required: (000), (001), (010), (011), (100), (101), (110), and (111).

B.QEA Structure



The structure of the QEA algorithm is as following.

Procedure QEA
begin
t = 0
i) initialize Q(t)
ii) make P(t) by observing the states of Q(t)
iii) evaluate P(t)
iv) store the best solutions among P(t) into B(t)
while (not termination-condition) do
Begin
t = t + 1
v) make P(t) by observing the states of Q(t - 1)
vi) evaluate P(t)
vii) update Q(t) using Q-gates
viii) store the best solutions among B(t - 1) and P(t) into B(t)
ix) store the best solution b among B(t)
x) if (migration-condition)
then migrate b or bjt to B(t) globally or locally, respectively
end
end

APPLICATION EXAMPLE

This section contains a full description of the QEA algorithm for the knapsack problem. To illustrate how QEA can be used to solve the combinatorial optimization problem, the knapsack problem is taken into consideration. Three different types of GA methods are briefly presented for comparison's sake. The "knapsack problem" can be defined as choosing the most lucrative things out of a variety of options given the knapsack's capacity limitations. The following is a description of the 0-1 knapsack problem. Choose a subset of the items to maximize the profit f(x), given a collection of m things and a knapsack.

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

Subject to:

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

Where $x = (x1, x2, \dots, xm)$, xi *is* 0 *or* 1, pi is the profit of item i, wi is the weight of item i, and C is the capacity of the knapsack. If $xi = 1$, *the ith item is selected for the knapsack.* QEA for the knapsack problem is a basic QEA algorithm with a repair process to satisfy the constraints of the capacity:

```
Procedure QEA for the Knapsack
Problem
begin
      t = 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 < MAX GEN) do
      begin
            t  = 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 b among B(t)
            if (migration-period)
            then migrate b or btj
            to B(t) globally or
            locally, respectively
      end
end
```

where the step make P(t) is implemented for each Q-bit individual to obtain the binary string x as follows:

```
Procedure Make (x)
begin
      i = 0
      while (i < m) do
      begin
            i = i + 1
            if  random[0,  1) |βi|²
            <
            then xi  = 1
            else xi  = 0
      end
end
```

The following repair procedure is used when the capacity constraint is broken by the binary string:

```
Procedure Repair (x)
begin
      knapsack-overfilled      =
      false
          m
      if  ∑ wixi > C
         i=1
      then knapsack-overfilled =
      true
      while
      (knapsack-overfilled) do
      begin
            select  an  ith  item
            from the knapsack
            xi = 0
                m
            if  ∑ wixi ≤ C
               i=1
            then
            knapsack-overfilled =
            false
      end
      while                  (not
      knapsack-overfilled) do
      begin
            select  a  jth  item
            from the knapsack
            xj = 1
                m
            if  ∑ wixi > C
               i=1
            then
            knapsack-overfilled =
            true
      end
      xj = 0
end
```

The following describes the Q-bit updating process:

```
Procedure Update (q)
Begin
      i = 0
      While (i<m) do
      begin
            i=i+1
```

```
        Determine  Δθi  with
        the  lookup  table
        obtain  (αi new, βi new)
        from the following:
              if  (  q  is
              located  in  the
              first/third
              quadrant)
              Then
[ αi new  βi new ]^T = U(Δθ_i) [ αi old  βi old ]^T
              Else
[ αi new  βi new ]^T = U(−Δθ_i) [ αi old  βi old ]^T
        End
q new = q old
end
```

A rotation gate is employed to update a Q-bit individual q as follows:

$$\begin{bmatrix} \alpha^i new \\ \beta^i new \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha^i old \\ \beta^i old \end{bmatrix}$$

The angle parameters used for the rotation gate in this knapsack problem are displayed in Table I.

Let us define an angle vector $\Theta = [\theta 1 \theta 2 \ldots \theta 8]^T$, where $\theta 1 \theta 2, \ldots, \theta 8$ can be selected easily through intuitive reasoning. For example, if xi *and* bi are 0 and 1, respectively, and if the condition $f(x) \geq f(b)$ is false:

i) if the Q-bit is located in the first or the third quadrant in Fig. 2, $\Delta\theta i$, the value of is set to a positive value to increase the probability of the state 1;

ii) if the Q-bit is located in the second or fourth quadrant, $\Delta\theta i$ should be used to increase the probability of state 1.

If xi and bi are 1 and 0, respectively, and if the condition $f(x) \geq f(b)$ is false:

i) if the Q-bit is located in the first or the third quadrant, $\Delta\theta i$ is set to a negative value to increase the probability of the state 0;

ii) if the Q-bit is located in the second or the fourth quadrant, $\Delta\theta i$ should be used to increase the probability of the state 0

.

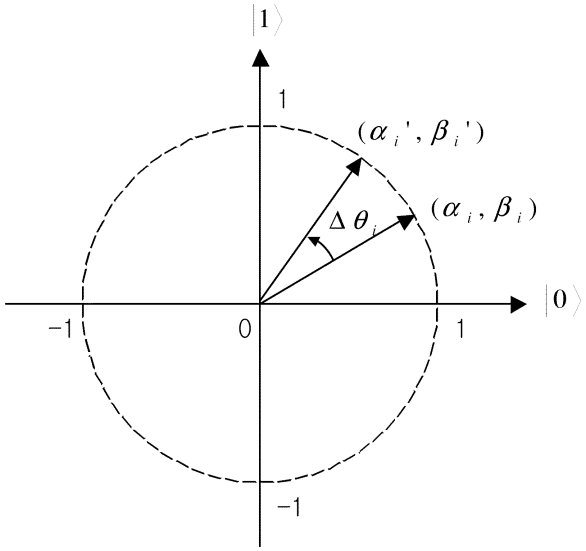


Fig. 2. Polar plot of the rotation gate for Q-bit individuals

TABLE I

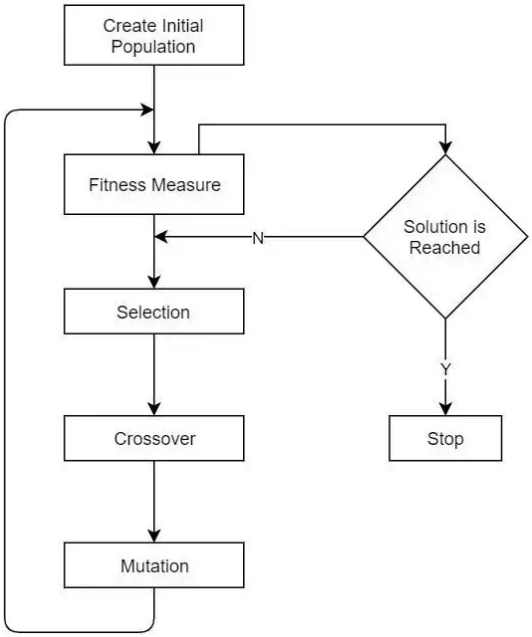
LOOKUP TABLE OF $\Delta\theta i$, WHERE f(.) IS THE PROFIT, AND *bi* AND *xi* ARE THE iTH BITS OF THE BEST SOLUTION b AND THE BINARY SOLUTION x, RESPECTIVELY. IN THE KNAPSACK PROBLEM, $\theta 1 = 0$, $\theta 2 = 0$, $\theta 3 = 0.01\pi$, $\theta 4 = 0$, $\theta 5 = \pi 0{,}01$, $\theta 6 = 0$, $\theta 7 = 0$, $\theta 8 = 0$ WERE USED

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

.

B. GA Methods for the Knapsack Problem

To see the performance of QEA I used standard GA to compare the results and the performance. The genetic algorithm that is used is as follows:



First, the initial population is declared. The concept behind chromosome encoding in this problem is to have a chromosome made up of as many genes as there are total items, with each gene index corresponding to a specific item index in the list. Each gene has a value of 1 or 0, indicating whether or not the relevant component is present.

We will use the following fitness function to solve this problem:

$$fitness = \sum_{i=1}^{n} c_i v_i; if \sum_{i=1}^{n} c_i w_i \leq kw$$

$$fitness = 0; otherwise$$

where,

n = chromosome length, c_i = ith gene, v_i = ith value , w_i = ith weight, kw = knapsack weight

Then, we pick the most physically fit people to go through crossover.

One-point crossover will be used for crossover. In order to ensure that the fittest individuals undergo crossover, we will set the crossover rate to a high value. In mutation, the chromosome that will change is chosen at random. Using the bit-flip method, we will change the selected gene that will undergo mutation from 1 to 0 and vice versa in order to create mutants.

Experimental Results and Analysis

In this experiment, the following data was created by this function:

$Wi = uniformly\ random[1, 10]$

$pi = wi + 5$

And the following capacity was used:

$$C = 1/2 \sum_{i=1}^{m} wi$$

The data were unsorted, and three data sets with 100, 250, and 500 items were considered from this site [3].

The experiment will be based on the number of iterations, and we will compare the mean fitness for both algorithms.

	Mean best fitness				
Algorithm	Genetic-Algorithm		Quantum-inspired Evolutionary-Algorithm		
Number of iterations	10	60	10	60	100
100	2600	3332	3221	3792	3942
250	7224	7689	8145	9253	10321
500	13756	14122	15936	19812	20875

As we see from the experiment, the QEA is more efficient than the GA, and it gives very good results when the number of iterations is larger.

Conclusion

This research put out a model QEA that was motivated by the idea of quantum computing. For the probabilistic representation, a string of Q-bits was designated as a Q-bit person. A Q-gate is created as a variation operator in order to introduce the variation to the Q-bit individual. The Q-bit representation for population diversity, the observation process that yields a binary string from a Q-bit individual, the update process that uses the Q-gate to push individuals toward better solutions, the migration process that allows for more variation in the Q-bit individuals' probabilities, and the termination condition that can be determined by the probability of the best solution are the characteristics of the proposed QEA. The knapsack issue was taken into consideration as a practical example for examining QEA's effectiveness.

Even with just one Q-bit person, QEA performed admirably in the experimental results. The straightforward knapsack problem, which only requires ten things, can be used to confirm the features of QEA. The outcomes showed how well QEA worked and how it might be applied to the combinatorial optimization issue.

References

[1] Kuk-Hyun Han and Jong-Hwan Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, Dec. 2002, doi: 10.1109/tevc.2002.804320.

[2] S. Tiwari, "Genetic Algorithm: Part 3 — Knapsack Problem," *Medium*, Apr. 28, 2019. https://medium.com/koderunners/genetic-algorithm-part-3-knapsack-problem-b59035ddd1d6

[3] M.B. Mohtasham, "Quantum-Evolutionary-Algorithm-Knapsack-Python-," *GitHub*, Sep. 20, 2022. https://github.com/mjBM/Quantum-Evolutionary-Algorithm-Knapsack-Python- (accessed Dec. 06, 2022).

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

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

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