Published in IET Control Theory and Applications
Received on 30th June 2011
Revised on 6 March 2012
doi: 10.1049/iet-cta.2011.0382

ISSN 1751-8644

Classical and quantum-inspired electromagnetism-like mechanism and its applications

Y.-H. Chou[1] C.-Y. Chen[2] C.-H. Chiu[1] H.-C. Chao[2,3]
[1]Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan
[2]Department of Electrical Engineering, National Dong Hwa University, Taiwan
[3]Department of Electronic Engineering and Institute of Computer Science and Information Engineering, National Ilan University, Taiwan
E-mail: hcc@niu.edu.tw

Abstract: In this study, we propose a novel evolutionary computing method that is called quantum-inspired electromagnetism-like mechanism (QEM). QEM is based on the electromagnetism theory and uses the characteristic of quantum computation. We compare with quantum-inspired evolutionary algorithms (QIEA) and the other traditional heuristic algorithms to solve 0/1 knapsack problem. Meanwhile, we can also find the shortest tour length of 16 cities travelling salesman problem (TSP). The experimental results show that the QEM can rapidly and efficiently obtain the optimal solution of combinatorial optimisation problems. To conclude, QEM is an efficient solution for optimal control and adaptive control in stochastic/hybrid systems.

1 Introduction

Evolutionary algorithms (EA) perform approximating solutions to all types of problems such as robotics and engineering applications [1, 2]. Quantum-inspired evolutionary algorithm (QIEA) is a special issue of estimation of distribution algorithm (EDA), where quantum probability amplitudes are used with probabilistic models to describe the promising areas of decision space and to guide the exploration of the global optimal. Different from other EDAs, it has more than one probability variables that can be used simultaneously. Besides, it enhances the traditional evolutionary computing as well [3–23].

The most successful case of using quantum-inspired evolutionary computing is by Han and Kim in 2002 [4]. In general, this method can be used to deal with some difficult problems. For example, the travelling salesman problem (TSP), knapsack problem, generalised assignment problem, etc. In this paper, we propose a novel evolutionary computing method which is called quantum-inspired electromagnetism-like mechanism (QEM) [24]. We first solved the 0/1 knapsack problem by using the classical electromagnetism-like (EM) algorithm. However, because of the intrinsic property of 0/1 knapsack problem and the mechanism of EM algorithm, we find the proposed QEM is more suitable than classical EM for solving the 0/1 knapsack problem. We also demonstrate the proposed QEM has good performance for deal with TSP.

The rest of the paper is organised as follows. Section 2 briefly introduces EM-like mechanism. Then Section 3 describes some basic characteristic of quantum computation. Follow that, a hybrid algorithm called QEM is proposed in Section 4. In Section 5, we will test some different benchmarks and present the experiment results of 0/1 knapsack problem and TSP. Finally, conclusions and discussions about QEM are made in Section 6.

2 EM-like mechanism

EM-like mechanism is a heuristic algorithm developed by Birbil and Fang in 2003 [5]. Like particle swarm optimisation (PSO) [25–27], EM is also a population-based algorithm. EM simulates charged points attracted or repulsed to each other in a feasible domain. By the electromagnetic forces between the points, it will finally converge to the highly attractive region and find the optimal solutions. The variables are bounded in the form of

$$\begin{aligned} &\text{Min.} \quad f(x) \\ &\text{s.t.} \quad x \in [l, u] \\ &\text{where} \quad [l, u] = \{x \in \Re^n \mid l_k \le x_k \le u_k, k = 1, 2, \ldots, n\} \end{aligned}$$

n is the dimension of the problem;
x is the possible solution of the problem;
l_k is the lower bound in the kth dimension;
u_k is the upper bound in the kth dimension; and
$f(x)$ is the object function.

In this n-dimensional solution space, each point x represents a solution and a charge q is associated with each point (or particle). The charge q is related to the objective function's value $f(x)$, which is associated with the solution point x. As in evolutionary search algorithms, a population, or set of solution, is created. Each solution point will exert attraction or repulsion on other points of which the magnitude of the electrostatic force is proportional to the product of the charges and inversely proportional to the square of the distance between the points (Coulomb's Laws). The principle behind the algorithm is that inferior solution points will prevent a move in their direction by repelling other solution points in the population, and that superior solution points will facilitate the moves in their direction.

EM is divided into four phases: (i) initialise the problem, (ii) local search the neighborhood points, (iii) calculate the force exerted on particles, and (iv) update the new position on every particle.

The general EM scheme is as shown in Fig. 1:

The *Initialise*() procedure randomly selects m points in the feasible domain, which is an n dimension hyper-cube. After the points are decided, it uses the object function to calculate the value of each point and the point with the best value will be stored in x^{best}. Then, the *Local*(LSITER,δ) procedure is used to local search the adjacent region of each point. Also, the number of LSITER and δ should be determined. LSITER and δ represent the number of the iterations and a multiplier for the local neighbourhood search, respectively. If we find any new point in the neighbourhood and the value of new point is better than x^{best}, it will be replaced until the number of LSITER is reached. After local search procedure, we will calculate the charges q of each point according to their object function values. The formula of charges is defined as

$$q^i = \exp\left(-n\frac{f(x^i) - f(x^{\text{best}})}{\sum_{j=1}^{m}\left(f(x^j) - f(x^{\text{best}})\right)}\right), \quad (1)$$

where x^{best} is the currently best solution to the object function and the variable n is the dimension of the problems. For more details of the definition of the parameters, please refer to [5].

After calculating the charges of each point by using object function, we can work out the forces between every two points. For any point i, we could know the resultant force exerted on the point by using

$$F^i \leftarrow F^i + (x^j - x^i)\frac{q^i q^j}{\|x^j - x^i\|^2}, \quad \forall j = 1, 2, \ldots, m; j \neq i \quad (2)$$

and

$$F^i \leftarrow F^i - (x^j - x^i)\frac{q^i q^j}{\|x^j - x^i\|^2}, \quad \forall j = 1, 2, \ldots, m; j \neq i \quad (3)$$

For any given i and j, if $f(x^j) < f(x^i)$, it means the magnetic force between these two points is an *attraction* as in (2); otherwise, it is *repulsion* as in (3). This is the main mechanism that moves the points around in feasible domain to find out the optimal solution. After that, the final step is *New_Position*(F). Using the resultant of forces computed previously, now we can update the new position of all points. The *New_Position*(F) is formulated as

$$x_k^i \leftarrow x_k^i + \lambda\frac{F^i}{\|F^i\|}(u_k - x_k^i), \quad \text{if } F^i > 0, \quad \begin{array}{l}\forall i = 1, 2, \ldots, m\\ \forall k = 1, 2, \ldots, n\end{array} \quad (4)$$

and

$$x_k^i \leftarrow x_k^i + \lambda\frac{F^i}{\|F^i\|}(x_k^i - l_k), \quad \text{if } F^i \leq 0, \quad \begin{array}{l}\forall i = 1, 2, \ldots, m\\ \forall k = 1, 2, \ldots, n\end{array} \quad (5)$$

where λ is a uniform random number between $[0, 1]$. An example of the attract–repulse effecting on three particles is shown in Fig. 2.

3 Quantum computing principles

The basic unit of information in computing and telecommunication in today's digital computers is one single bit, which is always in either the state '1' or '0' at any given time. The corresponding analogue on a quantum computer is represented by a quantum bit or *qubit* [28, 29]. Similar to classical bits, a qubit may be in the basis state '0' or '1' but additionally may also in any superposition of both states. Besides, the act of measuring (or observing) a qubit will project the quantum system onto one of its basis states [30, 31].

A quantum bit state $|\psi\rangle$ can be represented as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \quad (6)$$

where α and β are complex numbers, and $|\alpha|^2$ and $|\beta|^2$ are represented as the probability that the qubit will be

```
m: the number of population
MAXITER: the maximum number of iterations
LSITER: the maximum number of local search iterations
δ: local search parameter, δ ∈ [0, 1]
 1) Initialise()
 2) iteration = 1
 3) while iteration <= MAXITER do
 4)     Local(LSITER,δ)
 5)     F = Force()
 6)     New_Po ition(F)
 7)     iteration = iteration + 1
 8) End while
```

Fig. 1 *ALGORITHM: EM (m, MAXITER)*

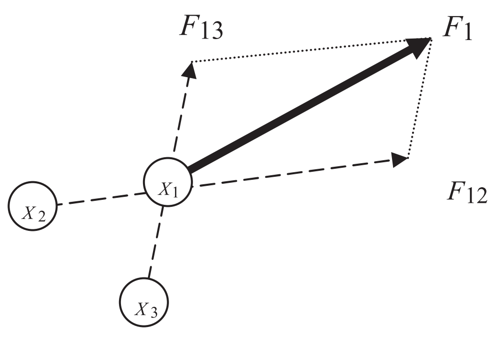

Fig. 2 *An example of attract-repulse effect on particle number X_1*

found in the '0' state and '1' state, respectively. The sum of $|\alpha|^2$ and $|\beta|^2$ will follow the rule of probability and can be represented as

$$|\alpha|^2 + |\beta|^2 = 1 \tag{7}$$

If there are n-quantum bits in a quantum system, we can represent all of them as

$$\left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \beta_1 & \beta_2 & \cdots & \beta_n \end{array}\right] \tag{8}$$

According to (7), for each $|\alpha|^2 + |\beta|^2 = 1$, $i = 1, 2, \ldots n$.

A key point when considering quantum systems is that they can compactly convey information on a large number of possible system states. In classical bit strings, a string of length n can only represent one of the 2^n possible states. However, a quantum system of n qubits can represent a total of 2^n dimensions. This means that even a short qubits can convey information on many possible system states. The representation of quantum bits as (8) is more easily to express any superposition of the states, and it is more suitable than other classified representations when applying to generate populations of EA.

The process of generating binary strings from the qubit string β is called measurement (observation). Let us have a binary solution x by observing the states of β, where $x = x_1, x_2, \ldots, x_n$. At any generation in the binary string of length n, x is formed by selecting either 0 or 1 for each bit by comparing the probability, either $|\alpha|^2$ or $|\beta|^2$, with R. In a quantum computer, the act of observing a quantum state will cause the qubit state to collapse into a basis state. However, collapsing into a single state does not occur in QEM, since QEM is working on a classical computer, not the quantum computer. In other words, this proposed QEM algorithm does not have to use any quantum computers for performing real quantum computation. That is the reason why we use the term of Quantum-inspired. Besides, to observe (measure) a Q-bit string β, a string R consisting of the same number of random numbers between 0 and 1 is generated, and the elements of string x is set to 0 if R is less than square of β; otherwise, x is set to 1. For example, Table 1 shows the observation process.

4 QEM

In this section, we describe the QEM algorithm and denote the variables used in the algorithm as shown in Fig. 3:

MAXITER: the maximum number of iterations.

β: the probability matrix of the quantum bits;
R: the reference matrix for qubits measurement;
x: the measurement result from β, possible solutions;
q: the charge of each particle; and
F: the electromagnetic force among particles.

The QEM steps are as follows:

Table 1 Observation of qubit string

Number of qubit	1	2	3	4	5	…	n
R	0.16	0.32	0.9	0.46	0.28	…	0.82
β	0.5	0.75	0.3	0.81	0.27	…	0.62
x	0	0	1	0	1	…	1

```
Proceure of QEM
Begin
    $iteration$=0
    initialise $\beta$
    while $iteration$ < MAXITER
    1) Generate $x$ by measuring $\beta$
    2) Repair $x$
    3) Evaluate $x$ and $f(x)$
    4) Calculate $q$ use the $f(x)$
    5) Calculate of total force vector $F$ globally by using $\beta$
       and $q$
    6) Update $\beta$ using the total force $F$
    7) $iteration$ +1
    End while
End
```

Fig. 3 *Procedure of QEM*

1. Create n items, m populations and R (with $U[0, 1]$). Then we measure β with R, when $R_k^i > \beta_k^i$, set $x_k^i = 1$, otherwise $x_k^i = 0$. β can be expressed as the following matrix

$$\beta = \begin{bmatrix} \beta_1^1 & \beta_1^2 & \cdots & \beta_1^m \\ \beta_2^1 & \beta_2^2 & \cdots & \beta_2^m \\ \vdots & \vdots & \ddots & \vdots \\ \beta_n^1 & \beta_n^2 & \cdots & \beta_n^m \end{bmatrix} \tag{9}$$

2. Compare those populations with the limit of boundary condition (capacity), and then repair x.
3. Calculate every profit of each x with the object function $f(x)$ and compute the maximum profit from all populations.
4. Calculate charge q using $f(x)$, which can determine the force vectors in the next step.
5. Use each of the q and β to calculate the total force vector F.
6. Finally, apply the force vector F to update and move the β in order to create the next generation. It is worthy to explain the operation of updating β. Because of these characteristics of qubit, when we increase the value of β, the probability of $|1\rangle$ will rise up. In the mean time, the value of α becomes lower; it means the probability of $|0\rangle$ will fall down. The physical meaning of updating β is that use a quantum rotation gate to change the value of β in physics.

The flowchart of the QEM is shown in Fig. 4.

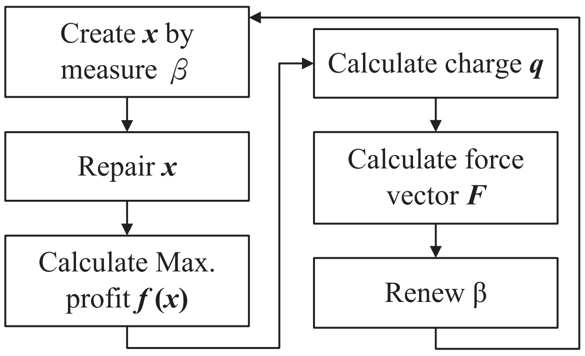
Create x by measure β
Repair x
Calculate Max. profit $f(x)$
Calculate charge q
Calculate force vector F
Renew β

Fig. 4 *Flowchart of QEM*

```
while (iteration $\leq$ MAXITER)
Begin
    for $k$=1 to $n$
        $R \in$ random number $U[0,1]$
        if $R > \beta$
            then set $x_k = 1$
        else  set $x_k = 0$
        end if
    end for
End while
```

a

```
Begin
    Set the knapsack-overfilled false
    if $\sum_{k=1}^{n} w_k x_k > capacity$
    for $k = 1$ to $n$
    then set the knapsack-overfilled true
    end if
    while (knapsack-overfilled) do
    Begin
        select an $i$-th item randomly from the knapsack
        $x_i \leftarrow 0$
        if $\sum_{k=1}^{n} w_k x_k < capacity$
          then set the knapsack-overfilled false
        end if
    end while
    while (no knapsack-overfilled) do
    Begin
        Select an $j$-th item randomly from the knapsack
        $x_j \leftarrow 1$
        if $\sum_{k=1}^{n} w_k x_k > capacity$
          then set the knapsack-overfilled true
        end if
    end while
    $x_j \leftarrow 0$
End
```

b

```
Begin
    for $i$=1 to $m$
        calculate $f(x^i)$ using $x$
    end for
    $f(x^{best}) \leftarrow \{f(x^i)\}$
End
```

c

Fig. 5 *QEM to solve the 0/1 knapsack problem*

a Procedure measure x
b Procedure repair x
c Procedure evaluate x
d Procedure calculate q
e Procedure calculate total force vector F
f Procedure renew β

```
Begin
    for $j$=1 to $m$
        $q^i = \exp\left(-n\frac{f(x^i)-f(x^{best})}{\sum_{j=1}^{m}\left(f(x^j)-f(x^{best})\right)}\right)$
    end for
End
```

d

```
$F^i \leftarrow 0$
Begin
    for $i$=1 to $m$
        for $j$=1 to $m$
            if $f(x^j) < f(x^i)$ then
                $F^i \leftarrow F^i + (x^j - x^i)\frac{q^i q^j}{\|x^j - x^i\|^2}$
            else
                $F^i \leftarrow F^i - (x^j - x^i)\frac{q^i q^j}{\|x^j - x^i\|^2}$
            end if
        end for
    end for
End
```

e

```
$\lambda \leftarrow U[0,1]$
Begin
    for $i$=1 to $m$
        for $k$=1 to $n$
            if $F_k^i > 0$
                $\beta_k^i \leftarrow \beta_k^i + \lambda\frac{F^i}{\|F^i\|}(u_k - x_k^i)$
            else
                $\beta_k^i \leftarrow \beta_k^i + \lambda\frac{F^i}{\|F^i\|}(x_k^i - l_k)$
            end if
        end for
    end for
End
```

f

Fig. 5 *continued*

5 Experimental results

5.1 Solving 0/1 knapsack problem

In this section, we will use the QEM to solve the 0/1 knapsack problem. The 0/1 knapsack problem is a well-known NP-complete problem. It can be described as: given a set of m items and a knapsack, select a subset of items to maximise the total profit $= \sum_{k=0}^{n} p_k x_k$, subject to $\sum_{k=0}^{n} w_k x_k \leq C$, where $\forall k$, $x_k = 0$ *or* 1, w_i is the weight of the ith item, p_i is the profit of the ith item, and C is the capacity of the knapsack. If the ith item is selected for the knapsack, $x_i = 1$, otherwise $x_i = 0$.

To the best of our knowledge, we have never found any previous literature that solves this problem by using EM algorithm. In the following experiment, we will first solve 0/1 knapsack by using EM algorithm. Because in classical EM algorithm, the value of x can only be set to either 0 or 1, which is always on their upper bound and lower bound, we can only regard $\|x^i - x^j\|^2$ as the hamming distance. Although EM seems not suitable for the 0/1 knapsack problem, we still can easily obtain better results than the classical genetic algorithm and they will be shown in the following tables and figures.

Furthermore, our QEM is implemented in a similar but different way. Fortunately, we have a parameter called β which can always move between 0 and 1. We think that is the main reason why QEM is more suitable than EM in solving 0/1 knapsack problem. The main parts of the pseudo-codes are listed as in Fig. 5*a*.

As in [4], we also use the same repair procedure as follows (see figure Figs. 5*b*–*f*):

It is noteworthy that the computation of $\|x^i - x^j\|^2$ is equal to $\sum(\beta^i - \beta^j)^2$ in quantum-inspired EM. Besides, a uniformly random number λ is also used as in [5]. λ is a random factor moving between 0 and 1. Finally, we can use the renewed β to evolve the next generation of populations.

In our experiments, we use the similar data sets as in [19]. The average knapsack capacity is used $C = Capacity = (1/2)\sum_{k=1}^{n} w_i$, the weights set to uniform random $w_i \in [1, 10]$, and the profits are equal to the sum of the weights and a uniform random number l_i as given by $p_i = w_i + l_i$, where $l_i \in [0, 5]$. We considered three different knapsack problem with 100, 250 and 500 items, and compared the proposed QEM with quantum-inspired genetic algorithm (QGA), classical EM-like algorithm and conventional genetic algorithm (CGA). In Table 2 and Figs. 6–8, the maximum number of the generation is 1000 and the number of runs is 30. The population size is 10 in both QEM and QGA. For CGA, we set the population size to 100. The probabilities of crossover and mutation are, respectively, fixed to 0.65 and 0.05 as in [20]. The experimental results show that in the same setting of all three types of different item numbers, the best profits, the mean profits and the worst profits of QEM are much better than those of QGA, EM and CGA algorithms.

In addition, we compared QEM with other traditional heuristic algorithms, such as the hill-climbing algorithm (HILL), simulated annealing algorithm (SA) and tabu search algorithm (TABU). All methods are tested for 10 times and the generation is set to 1000. The results are shown that QEM is superior to the others in Table 3 and Figs. 9–11. We also perform experiment of QEM with different population size as 10, 20 and 30, respectively. The results are shown in Table 4 and Figs. 12–14, which are the average of 30 tests, and generation is set to 1000.

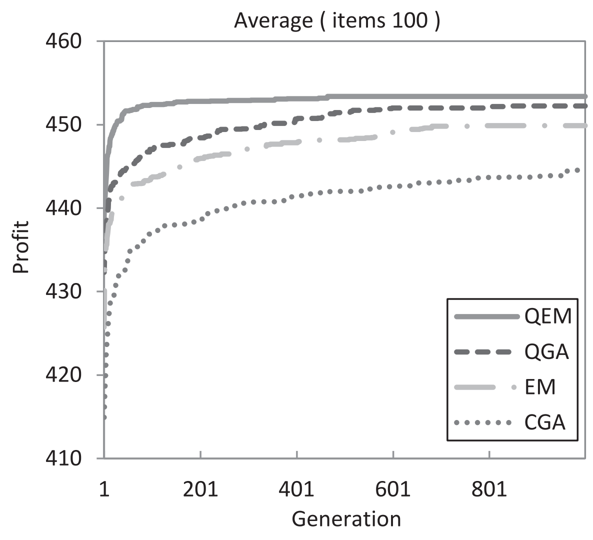

Fig. 6 *Comparison on the knapsack problem with 100 items*

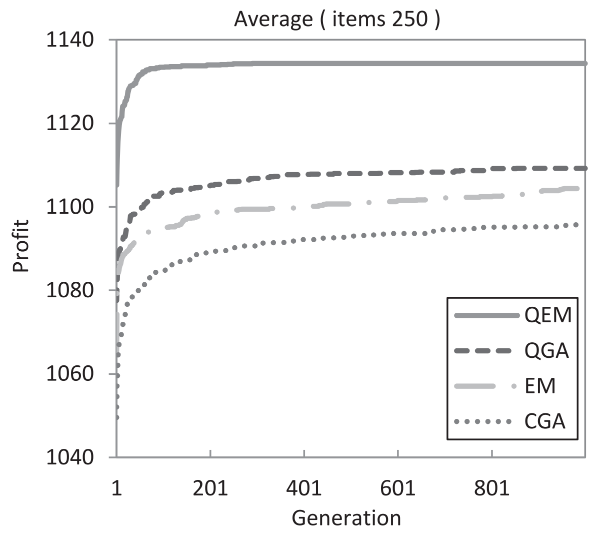

Fig. 7 *Comparison on the knapsack problem with 250 items*

Table 2 Experimental results of the 0/1 knapsack problem

Number f items			CGA	EM	QGA	QEM
100	profits	best	467.5	472.5	477.7	**492.6**
		mean	444.5	449.9	452.2	**453.4**
		worst	414.9	425.7	432.3	**435.9**
t(s/run)			0.001073	0.05915	0.016852	0.00579
250	profits	best	1138.0	1143.4	1141.2	**1201.4**
		mean	1095.7	1104.5	1109.2	**1134.3**
		worst	1049.6	1065.5	1077.6	**1105.2**
t(s/run)			0.001625	0.110153	0.060588	0.03304
500	profits	best	2253.8	2256.3	2257.9	**2289.4**
		mean	2161.1	2182.1	2182.8	**2186.6**
		worst	2089.1	2138.6	2134.4	**2147.0**
t(s/run)			0.002561	0.193722	0.153577	0.39987

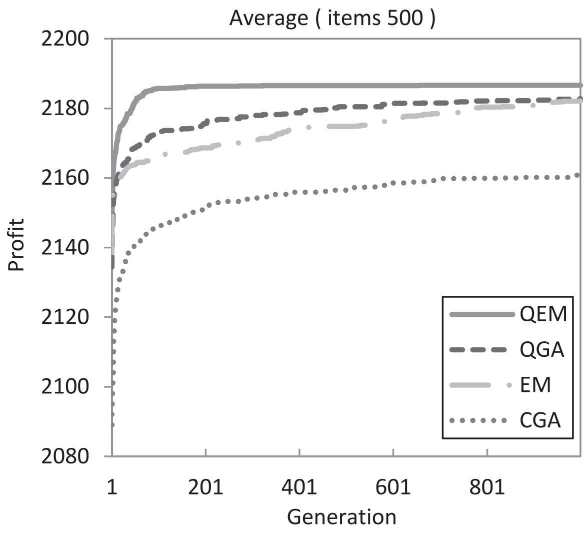

Fig. 8 *Comparison on the knapsack problem with 500 items*

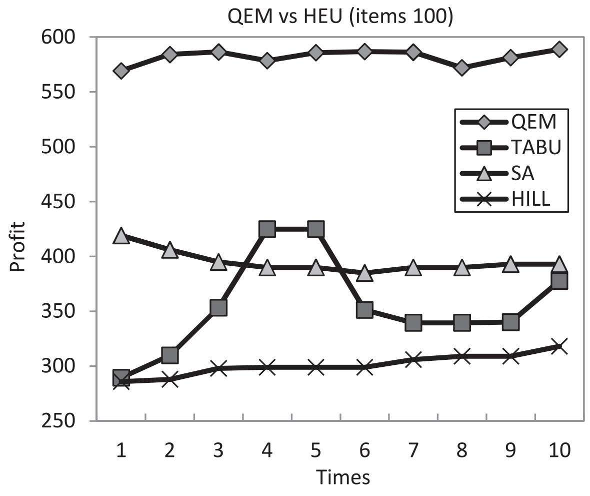

Fig. 9 *Comparison on the knapsack problem with 100 items*

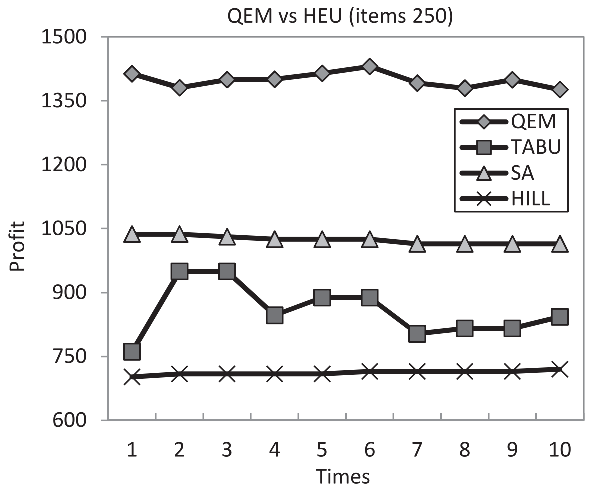

Fig. 10 *Comparison on the knapsack problem with 250 items*

Obviously, the larger the population size, the more profit and the faster convergent speed will be obtained. The above experiments are implemented in Matlab 7.6. The test environment is set up on a personal computer with Pentium E5300 CPU at 2.6 GHz CPU, 4G RAM and running on Windows XP.

5.2 Solving TSP

In this section, we will explain how to use QEM for solving TSP. TSP problem can be stated as follows: given a set of n cities $C = \{c_1, c_2, \dots, c_n\}$ and distances $D = \{(c_i, c_j) : c_i, c_j \in C, i \neq j\}$ for each pair of cities, the goal is to find the shortest tour that visits each city exactly once and then returns to the starting city. TSP is one of the well-known NP-hard combinatorial optimisation problems. These problems are widely considered unsolvable by using polynomial time algorithm. We consider that each particle of attraction or repulsion for finding the optimal solution is more suitable for solving TSP. The procedure of QEM for solving TSP as shown in Fig. 15.

The procedure of evaluate x are modified as follow (see Fig. 16).

The matrix β expressed as (9), where n is the number of cities and m is population size. Here we do not need to obtain X by measuring β, but by sorting β to obtain visited sequence $X = \{x^1, x^2, \dots, x^m\} x^i = c_1 \to c_2 \to \cdots \to c_n$. Then we evaluate total length of each sequence x^i with the object function $f(x)$, where $d(c_i, c_j)$ denotes the distance between cities c_i and c_j , and find the best objective value $f(x^{\text{best}})$.

The current feasible solution in the neighbourhood may find better solution. Therefore, we apply SA algorithm as local search algorithm. Viewing those solutions as points that can move left and right in feasible domain tries to seek better solutions around those possible solutions. First, we set the initial temperature to 100 and the temperature coefficient to 0.7. Exchange two cities randomly for all solutions, if new solution is better than the original solution, using new solution replace the old solution. Otherwise, decide to replace the original solution or not by random and reset temperature by using the following formula : *temperature* = *temperature* $\times$ 0.7, repeat the foregoing steps until *temperature* less than 1.

Table 3 Results of QEM against traditional heuristic algorithms in knapsack problem

Item number	100		250		500	
Method	Best profit	Time, s	Best profit	Time, s	Best profit	Time, s
QEM	588.81	6.41	1430.51	7.82	2825.34	20.36
HILL	318	3.13	720	3.50	1446	6.76
SA	419	6.25	1037	20.64	2057	25.37
TABU	424.80	8.34	949.32	17.63	1806.42	28.31

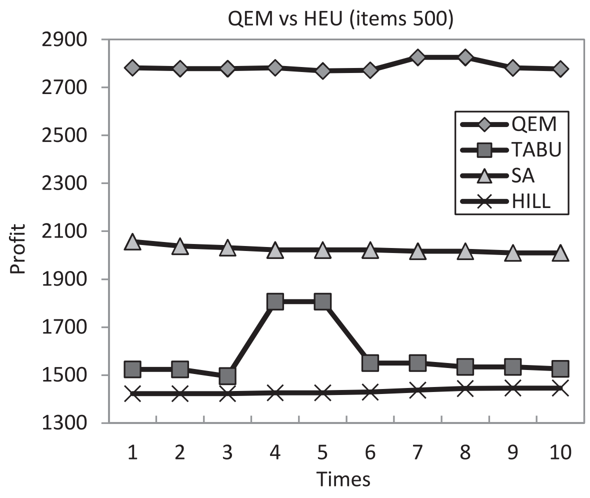

Fig. 11 *Comparison on the knapsack problem with 500 items*

The *Local_Search*() in Procedure of QEM for solving TSP as follow, where T represents the control temperature, and $PR(A)$ is the Boltzman's function as the probability of acceptance (see Fig. 17).

Using the *Local_Search*() can avoid falling into the local optimal solution. Besides, it enhances convergence properties. In the next step, the remaining procedures used to solve the TSP are just like the same as in 0/1 knapsack problem.

The experiment of the QEM for TSP is examined by the benchmark problem ULYSSES16 with 16 cities from the TSPLIB [32]. The data for the symmetric TSP and the geographical distance of 16 cities are shown in Table 5. The related parameter for QEM such as maximum number of the generation is set to 1000 and the population size is set to 100. Figs. 18–20 show the results, which the cost at first generation up to 9553. After about 500 generations, we could obtain the minimum cost with 6747, which is the best-known solution published from TSPLIB. One of the visited sequence is $3 \to 2 \to 4 \to 8 \to 1 \to 14 \to 13 \to 12 \to 7 \to 6 \to 15 \to 5 \to 11 \to 9 \to 10 \to 16$ shown in Fig. 19.

Finally, we calculate the search space to analyse the performance. All the solution space for the benchmark problem (ULYSSES16) is $16!/(16*2) = 653837184000$. In the QEM, the scale of the search space is $16*100*12*1000 = 19200000$, which is the product of the number of cities, population size and the running iterations. The search space of QEM is only 0.0029% of the solution space, which show the proposed algorithm could deal with the small-scale TSP efficiently and effectively.

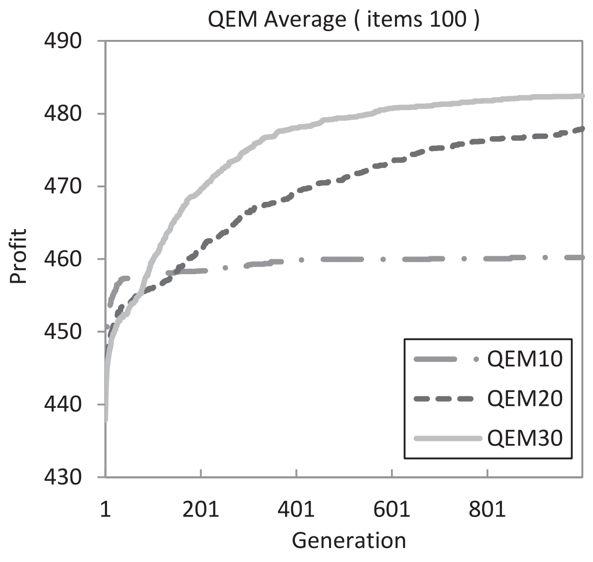

Fig. 12 *Comparison on the knapsack problem with 100 items*

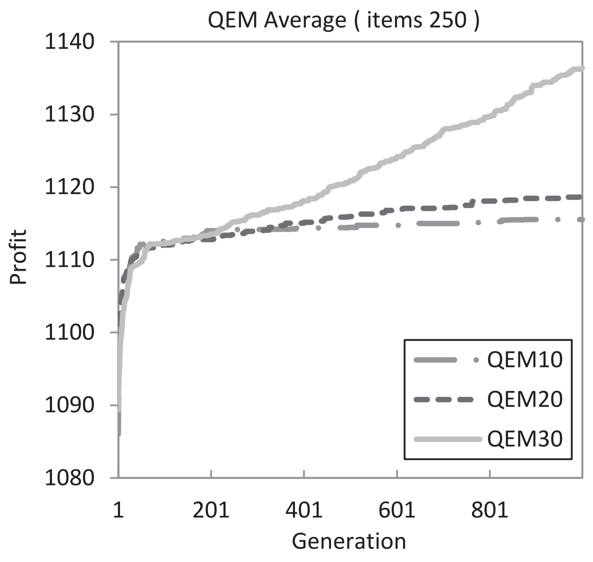

Fig. 13 *Comparison on the knapsack problem with 250 items*

Table 4 Results of different population size and generation of QEM in knapsack problem

Item number		100		250		500	
Size	Generation	Best profit	Time, s	Best profit	Time, s	Best profit	Time, s
10	100	457.76	0.18	1112.55	0.26	2186.92	0.42
	500	459.95	0.80	1114.44	1.09	2189.77	1.81
	1000	460.20	1.61	1115.53	2.64	2191.30	3.87
20	100	455.97	0.51	1112.02	0.78	2198.33	1.26
	500	471.06	2.19	1115.95	3.76	2202.69	5.65
	1000	477.95	4.53	1118.71	7.57	2204.06	12.15
30	100	459.74	0.95	1112.23	1.41	2203.59	2.19
	500	479.40	4.23	1120.85	6.75	2207.38	11.52
	1000	482.44	8.15	1136.38	12.78	2210.65	22.91

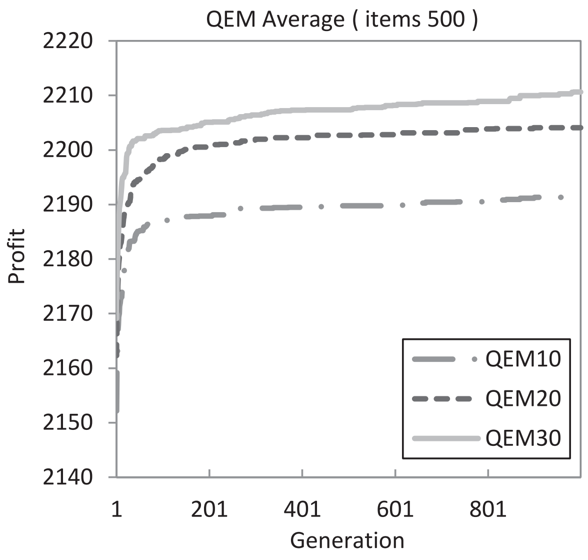

Fig. 14 *Comparison on the knapsack problem with 500 items*

```
Begin
    $iteration$=0
    initialise $\beta$
    while $iteration$ < MAXITER
  1) Obtain visited sequence $X$ by sorting $\beta$
  2) Evaluate $x$ and $f(x)$
  3) $Local\_Search()$
  4) Calculate $q$ use the $f(x)$
  5) Calculate of total force vector $F$ globally by using $\beta$
     and $q$
  6) Update $\beta$ using the total force $F$
  7) $iteration$ +1
    end while
End
```

Fig. 15 *Proceure of QEM for solving TSP*

```
Begin
    for $i$=1 to $m$
        $f(x^i) = -\sum_{j=1}^{n-1} d(c_j, c_{j+1}) + d(c_n, c_1)$
    end for
    $f(x^{best}) \leftarrow \{f(x^i)\}$
End
```

Fig. 16 *Procedure Evaluate x for solving TSP*

```
Begin
    initialise $T$=100
    while $T > 1$
        for $i$=1 to $m$
            $x^{temp} \leftarrow$ Select two cities to exchange from $x^i$
            if $f(x^{temp}) > f(x^i)$
                $f(x^i) = f(x^{temp})$
            else
                $PR(A) = min\{1, e^{\frac{-\left(f(x^{temp}) - f(x^i)\right)}{T}}\}$
                $R \in$ random number $U[0,1]$
                if $R < PR(A)$
                    $f(x^i) = f(x^{temp})$
                end if
            end if
        end for
        $T = T \times 0.7$
    end while
End
```

Fig. 17 *Procedure of Local_Search()*

Table 5 The data for the symmetric TSP of 16 cities

City position	*X*	*Y*
1	38.24	20.42
2	39.57	26.15
3	40.56	25.32
4	36.26	23.12
5	33.48	10.54
6	37.56	12.19
7	38.42	13.11
8	37.52	20.44
9	41.23	9.1
10	41.17	13.05
11	36.08	−5.21
12	38.47	15.13
13	38.15	15.35
14	37.51	15.17
15	35.49	14.32
16	39.36	19.56

6 Conclusions

In this paper, we proposed a novel evolutionary computing method which is called QEM to solve 0/1 knapsack problem and asymmetric TSP. We first solved the 0/1 knapsack problem by using the classical EM-like algorithm. However, because of the intrinsic property of 0/1 knapsack problem and the mechanism of EM algorithm, QEM is more suitable than classical EM for solving the 0/1 knapsack problem. In addition, QEM solved the benchmark problem ULYSSES16 to obtain an optimal distance value 6747. It demonstrated QEM that has good performance for deal with the small-scale TSP. QEM is based on the electromagnetism theory and utilises the characteristic of quantum computing. It can rapidly and efficiently obtain the optimal solution of the combinatorial optimisation problem. We compared our QEM with the CGA, QGA and classical EM. Experimental results show that QEM can effectively enhance the searching efficiency and improve the optimisation performance for combinatorial optimisation problems. As a result, the proposed QEM has the automatic balance ability between exploration and exploitation. Unlike most quantum-inspired algorithms manipulating the angles of rotations, a multi-dimensional way is used to improve the optimisation performance. Without using the local search method, we

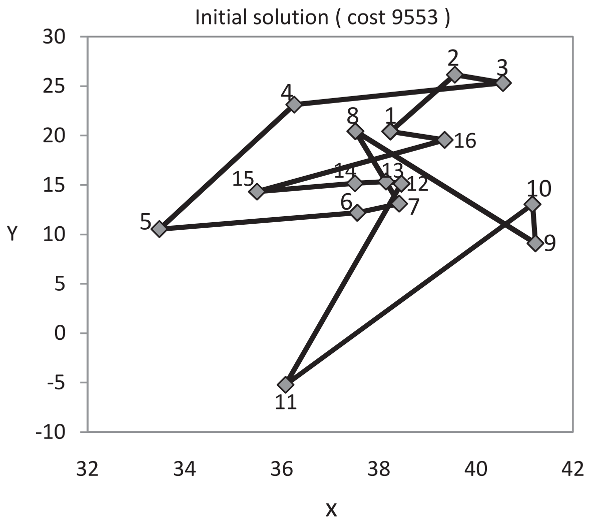

Fig. 18 *Results of optimal solution in Ulysses16*

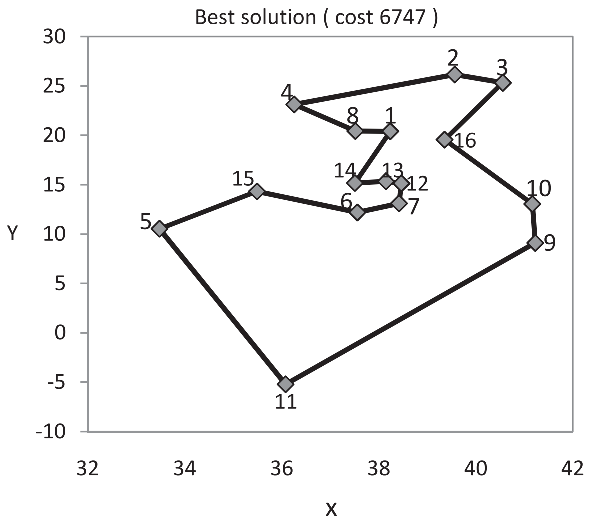

Fig. 19 *Results of optimal solution in Ulysses16*

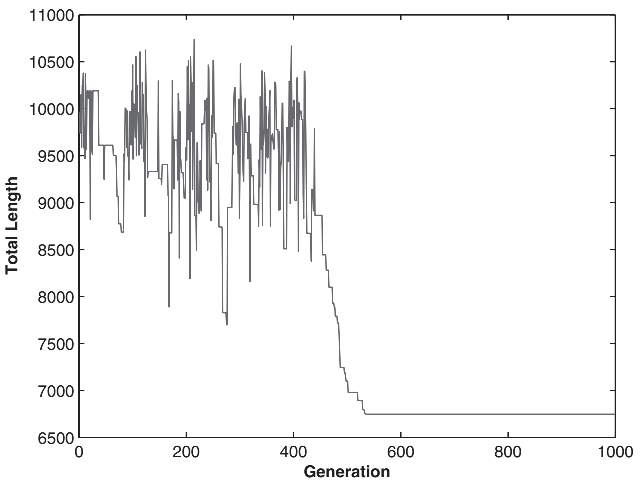

Fig. 20 *Results of optimal solution in Ulysses16*

have just utilised the probability of $U[0, 1]$ to generate each population. It shows that the results of experiments are still good enough and very competitive.

7 References

1 Hsu, C.-C., Chang, S.-C., Yu, C.-Y.: 'Tolerance design of robust controllers for uncertain interval systems based on evolutionary algorithms', *IET Control Theory Appl.*, 2007, **1**, (1), pp. 244–252

2 Dogan, M., Istefanopulos, Y.: 'Optimal nonlinear controller design for flexible robot manipulators with adaptive internal model', *IET Control Theory Appl.*, 2007, **1**, (3), pp. 770–778

3 Narayan, A., Patvardhan, C.: 'A novel quantum evolutionary algorithm for quadratic Knapsack problem'. IEEE Int. Conf. on Systems, Man and Cybernetics, SMC 2009, 11–14 October 2009, pp. 1388–1392.

4 Han, K.-H., Kim, J.-H.: 'Quantum-inspire evolutionary algorithm for a class of combinatorial,' *IEEE Trans. Evol. Comput.*, 2002, **6**, pp. 580–593.

5 Birbil, S.I., Fang, S.-H.: 'An electromagnetism-like mechanism for global optimization', *J. Glob. Optim.*, 2003, **25**, pp. 263–282

6 Han, K.-H., Kim, J.-H.: 'Quantum-inspire evolutionary algorithms with a new termination criterion, h_c gate, and two phase scheme', *IEEE Trans. Evol. Comput.*, 2004, **8**, (2), pp. 156–169

7 Debels, D., Vanhoucke, M.: 'An electromagnetism meta-Heuristic for the resource-constrained project scheduling problem'. (LNCS, **3871**), pp. 259–270 (Springer, 2006).

8 Holland, J.H.: 'Adaptation in natural and artificial systems', (University Michigan Press, Ann Arbor, MI, 1975)

9 Mahdabi, P., Jalili, S., Abadi, M.: 'A multi-start quantum-inspired evolutionary algorithm for solving combinatorial optimization problems'. Proc. Tenth Annual Conf. on Genetic and Evolutionary Computation, CECCO'08, Atlanta, GA USA, July 12–16, 2008, pp. 613–614.

10 Grover, L.K.: 'Quantum mechanical searching'. Proc. 1999 Congress on Evolutionary Computation, Piscataway, NJ, July 1999, **3**, pp. 2255–2261

11 Feynman, R.: 'Simulating physics with computers', *Int. J. Theor. Phys.*, 1982, **21**, (6), pp. 467–488

12 Destsch, D.: 'Quantum theory, the Church–Turing principle and the universal quantum computer', *Proc. R. Soc. London A*, 1985, **400**, pp. 97–117

13 Benioff, P.: 'The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines', *J. Stat. Phys.*, 1980, **22**, pp. 563–591

14 Deutsch, D.: 'Quantum computational networks', *Proc. R. Soc. London A*, 1989, **425**, pp. 73–90

15 Shor, P.W.: 'Algorithms for quantum computation: discrete logarithms and factoring'. Proc. 35th Annu. Symp. Foundations Computer Science, Sante Fe, NM, November 1994, pp. 124–134.

16 Shor, P.W.: 'Quantum computing', *Documenta mathematica*, vol. Extra Volume (ICM, 1998), pp. 467–486

17 Grover, L.K.: 'A fast quantum mechanical algorithm for database search'. Proc. 28th ACM Symp. on Theory Computing, 1996, pp. 212–219

18 Grover, L.K.: 'Quantum mechanics helps in searching for a needle in a haystack', *Phys. Rev. Lett., Amer. Phys. Soc.*, 1994, **79**, (2), pp. 325–328

19 Wanga, Y., Fenga, X.-Y., Huanga, Y.-X. *et al.*: 'A novel quantum swarm evolutionary algorithm and its applications', '*Neurocomputing*', January, 2007, pp. 633–640.

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

21 Li, Y., Zhao, J., Jiao, L., Wu, Q.: 'Quantum-inspired evolutionary multicast algorithm'. Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, SMC 2009, 11–14 October, 2009, pp. 1496–1501.

22 Jiao, L.C., Li, Y.Y., Gong, M.G., Zhang, X.R.: 'Quantum-inspired immune clonal algorithm for global optimization', *IEEE Trans. Syst. Man Cybern. B*, 2008, **38**, (5), pp. 1234–1253

23 Li, B.B., Wang, L.: 'A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling', *IEEE Trans. Syst. Man Cybern., B*, 2007, **37**, (3), pp. 576–591

24 Chou, Y.H., Chang, C.C., Chiu, C.H., Lin, F.J., Yang, Y.J., Peng, Z.Y.: 'Classical and quantum-inspired electromagnetism-like mechanism for solving 0/1 knapsack problems'. Proc. 2010 IEEE Int. Conf. on Systems Man and Cybernetics, SMC 2010, 10–13 October, 2010, pp. 3211–3218.

25 Sun, J., Feng, B., Xu, W.: 'Particle swarm optimization with particles having quantum behavior'. Proc. of Congress on Evolutionary Computing, CEC2004, 2004, vol. 1, pp. 325–331

26 Mikki, S., Kishk, A.: 'Investigation of the quantum particle swarm optimization for electromagnetic applications'. IEEE Antennas Propag. Soc. Int. Symp., 2005, vol. 2A, pp 45–48

27 Guo, W.Z., Xiong, N.X., Lee, C., Yang, L.T., Chen, G.L., Weng, Q.: 'Task allocation algorithm based on particle swarm optimization in heterogeneous computing environments', *J. Internet Technol.*, 2010, **11**, (3), pp. 343–351

28 Chen, C.Y., Chou, Y.H., Chao, H.C.: 'Distributed quantum entanglement sharing model for high-performance real-time system', Soft Computing, published online, May 2011.

29 Chou, Y.H., Chen, C.Y., Chao, H.C., Park, J.H., Fan, R.K.: 'Quantum entanglement and non-locality based secure computation for future communication', *IET Inf. Sec.*, 2011, **5**, (1), pp. 69–79

30 Dong, D., Lam, J., Tarn, T.J.: 'Rapid incoherent control of quantum systems based on continuous measurements and reference mode', *IET Control Theory Appl.*, 2009, **3**, (2), pp. 161–169

31 Dong, D., Petersen, I.R.: 'Quantum control theory and applications: a survey', *IET Control Theory Appl.*, 2010, **4**, (12), pp. 2651–2671

32 Standard TSPLIB, 1997. Online available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.