A COMPARATIVE STUDY OF THE CANONICAL GENETIC ALGORITHM AND A REAL-VALUED QUANTUM-INSPIRED EVOLUTIONARY ALGORITHM

KAI FAN

School of Business & Natural Computing Research and Applications Group, University College Dublin

Carysfort Avenue, Blackrock, Dublin, Ireland

kai.fan@ucd.ie

http://ncra.ucd.ie

ANTHONY BRABAZON

School of Business & Natural Computing Research and Applications Group, University College Dublin

UCD Complex Adaptive Systems Laboratory, Belfield Office Park, Belfield, Dublin4, Ireland

anthony.brabazon@ucd.ie

CONALL O'SULLIVAN

School of Business & Natural Computing Research and Applications Group, University College Dublin

Carysfort Avenue, Blackrock, Dublin, Ireland

conall.osullivan@ucd.ie

MICHAEL O'NEILL

School of Computer Science and Informatics & Natural Computing Research and Applications Group, University College Dublin

UCD Complex Adaptive Systems Laboratory, Belfield Office Park, Belfield, Dublin4, Ireland

m.oneill@ucd.ie

Received (Day Month Year) **15 / 09 / 2008**
Revised (Day Month Year) **05 / 01 / 2009**
Accepted (Day Month Year)

Purpose - Following earlier claims that Quantum-inspired Evolutionary Algorithm (QIEA) may offer advantages in high dimensional environments, this paper tests a real-valued QIEA on a series of benchmark functions of varying dimensionality in order to examine its scalability within both static and dynamic environments.
Design/Methodology/Approach – This study compares the performance of both the QIEA and the canonical genetic algorithm on a series of test benchmark functions.
Findings - The results show that the QIEA obtains highly competitive results when benchmarked against the genetic algorithm within static environments, while substantially outperforming both binary and real-valued representation of the genetic algorithm (GA) in terms of running time. Within dynamic environments, the QIEA outperforms GA in terms of stability and run time.
Originality/value - This study suggests that QIEA has utility for real-world high dimensional problems, particularly within dynamic environments, such as that found in real-time financial trading.

Keywords Estimation of Distribution Algorithm (EDA); Quantum-inspired evolutionary algorithm (QIEA); Quantum chromosome.

Paper type **Research paper**

1. Introduction

This paper describes a recently introduced algorithm, the quantum-inspired evolutionary algorithm (QIEA), and compares the performance of the QIEA on a series of benchmark functions with that of the genetic algorithm. This comparative study provides further insights into the performance of the QIEA and clearly illustrates its scaling and efficiency potential within both static and dynamic environment.

Over the last fifteen years a substantial literature has built up in the field of evolutionary computation (EC) concerning estimation of distribution algorithms (EDAs). EDAs are an alternative way of representing the learning which is embedded in evolving populations of genotypes in evolutionary computing (EC). EDAs have several names including *probabilistic model building algorithms* (PBMAs) and *iterated density estimation evolutionary algorithms* (IDEAs) (see (Larranaga, 2001, [15]), (Pelikan, 2005, [20]), (Pelikan, 2006, [21])). Recent years have seen the application of EDAs to most traditional evolutionary computing problem domains including multi-objective optimization (see (Thierens, 2001, [24]) and (Khan, 2002, [14]) and dynamic optimization (see (Yang, 2005, [27]).

Rather than maintaining a population of solution encodings from one generation to the next and manipulating this population using selection, crossover and mutation operators (as is the case in typical evolutionary algorithms), global statistical information is extracted from previous iterations of the EDA. This information is used to construct a posterior probability distribution model of

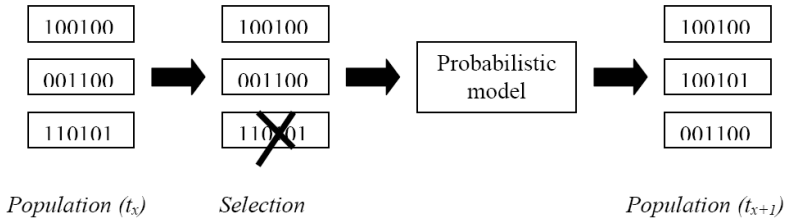


Fig. 1. Illustration of a binary-representation EDA, with sampling from a probabilistic model replacing the usual GA crossover and mutation operators.

promising solutions, based on the extracted information. New solutions are then sampled from this probability model (Fig. 1). Hence, EDAs maintain the selection and variation concepts from evolutionary algorithms but generate variation in a different way. The general EDA concept can be operationalised in multiple ways. For example, the model update step can be performed in different ways depending on the assumptions made concerning the problem. Examples of EDAs include population-based incremental learning (PBIL) [1], the univariate marginal distribution algorithm (UMDA) [16], mutual information maximization for input clustering (MIMIC) [13], factorized distribution algorithm (FDA) [17], the compact genetic algorithm (cGA) [12], and the Bayesian Optimization Algorithm (BOA) [19]. A simplified flowchart of a univariate EDA is as follows:

1. Initialize a probability vector P of length l (assume an l-dimensional problem)
2. Repeat
3. Generate n trial solution vectors, using P
4. Evaluate the n trial solutions
5. Select $x<n$ better solutions from the population
6. Adapt P using these x solutions
7. Test terminating condition

In Section 2 we provide a detailed introduction to quantum-inspired evolutionary algorithms (QIEA) but it is notable that most real-valued implementations of the QIEA bear some similarity to the EDA

paradigm. QIEAs differ from EDAs in terms of their inspiration with QIEAs being inspired by the mechanisms of quantum mechanics. Quantum mechanics is an extension of classical mechanics, modelling behaviors of natural systems observed particularly at very short time or distance scales. An example of such a system is a sub-atomic particle, such as a free electron. A complex-valued (deterministic) function of time and space co-ordinates, called the *wave-function*, is associated with the system: it describes the *quantum state* the system is in. The standard interpretation of quantum mechanics is that this abstract wave-function allows us to calculate probabilities of outcomes of concrete experiments. The squared modulus of the wave-function is a probability density function (PDF): it describes the probability that an observation of, for example, a particle will find the particle at a given time in a given region of space. The wave-function satisfies the *Schrödinger equation*. This equation can be thought of as describing the time evolution of the wave-function --- and so the PDF --- at each point in space: as time goes on, the PDF becomes more "spread out" over space, and our knowledge of the position of the particle becomes less precise, until an observation is carried out; then, according to the usual interpretation, the wave-function "collapses" to a particular classical state (or *eigenstate*), in this case a particular position, and the spreading out of the PDF starts all over again.

Before the observation we may regard the system as being in a linear combination of all possible classical states (this is called a *superposition of states*); then the act of observation causes one such classical state to be chosen, with probability given by the PDF. Note that the wave function may interfere with itself (for example, if a barrier with slits is placed in the "path" of a particle) and this interference may be constructive or destructive, that is, the probability of detecting a particle in a given position may go up or go down.

In this paper, we test the scalability and efficiency of the QIEA using a series of well-known benchmark problems drawn from the evolutionary computation literature. We initially introduce the real-valued Quantum-inspired Evolutionary Algorithm in Section 2. We then outline the experimental design adopted in Section 3. Following this, the results are provided in Section 4 and finally, in Section 5, a number of conclusions are drawn and some directions for future work are outlined.

2. Quantum-inspired Evolutionary Algorithm

Quantum mechanics is an extension of classical mechanics which models behaviors of natural systems that are observed particularly at very short time or distance scales. Under a quantum representation, the basic unit of information is no longer a bit which can assume two distinct states (0 or 1), but is a quantum system. Hence, a *qubit* (the smallest unit of information in a two-state quantum system) can assume either of the two ground states (0 or 1) or any superposition of the two ground states (the quantum superposition). A qubit can therefore be represented as

$$\left|q^i\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle \tag{1}$$

where $\left|0\right\rangle$ and $\left|1\right\rangle$ are the ground states 0 and 1, α and β are complex numbers with $\alpha^2 + \beta^2 = 1$, that specify the probability amplitudes of the two ground states. The act of observing (or measuring) a qubit projects the quantum system onto one of the ground states. The term α^2 is the probability that the qubit will be in state 0 when it is observed, and β^2 is the probability that it will be in state 1. Hence, a qubit encodes the *probability* that a specific ground state will be seen when an observation takes place, rather than encoding the ground states themselves. In order to ensure this probabilistic interpretation remains valid, the values for α and β are constrained such that $\alpha^2 + \beta^2 = 1$.

In recent years, quantum-inspired concepts have been applied to the domain of evolutionary computation [18; 10; 11; 25; 26]. Because QIEAs use a quantum representation, it has been suggested that they can maintain a good balance between exploration and exploitation, and that they offer computational efficiencies as use of a quantum representation can allow the use of smaller population sizes than typical evolutionary algorithms [6; 7; 8].

2.1. *Real-valued quantum-inspired evolutionary algorithms*

In the initial literature which introduced the QIEA, a binary representation was adopted, wherein each quantum chromosome was restricted to consist of a series of 0s and 1s. The methodology was modified to include real-valued vectors by da Cruz et al. [5]. As with binary-representation QIEA, real-valued QIEA maintains a distinction between a quantum population and an observed population of, in this case, real-valued solution vectors. However the quantum individuals have a different form to those in binary-representation QIEA. The quantum population *Q(t)* is comprised of *N* quantum individuals (q_i : i = 1,2,3,…, N), where each individual i is comprised of G genes (g_{ij} : j = 1,2,3,…, G). Each of these genes consist of a pair of values $q_{ij} = (g_{ij}, \sigma_{ij})$, where $g_{ij}, \sigma_{ij} \in \Re$ represent the mean and the width of a square pulse (Fig. 2). Representing a gene in this manner has a parallel with the quantum concept of superposition of states as a gene is specified by a range of possible values, rather than by a single unique value.

The original QIEA algorithms, e.g. [10; 11], are based very closely on physical qubits, but the "quantum-inspired" algorithm of da Cruz et al. [5] used in this paper draws less inspiration from quantum mechanics since it:

- does not use the idea of a quantum system (in particular, no qubits);
- only allows for constructive (not destructive) interference, and that interference is among "wave-functions" of *different* individuals;
- uses real numbers as weights, rather than the complex numbers which arise in superposition of states in physical systems;
- the probability density functions (PDFs) used (uniform distributions) are not those arising in physical systems.

However, the da Cruz et al algorithm does periodically sample from a distribution to get a "classical" population, which can be regarded as a wave-function (quantum state) collapsing to a classical state upon observation.

Quantum-inspired evolutionary algorithm

```
==================================================================
    Set t = 0
    Initialise Q(t) of N individuals with G genes
   while t < max t do
    Create the PDFs (and corresponding CDFs, see equation 4) for each gene locus
       using the quantum individuals
    Create a temporary population, denoted E(T), of K real-valued solution vectors
       by observing Q(t) (via the CDFs)
    if t = 0 then
          C(t) = E(t)
       (Note: the population C(T) is maintained between iterations of the algorithm)
    else
          E(t) = Outcome of crossover between E(t) and C(t)
       Evaluate E(t)
       C(t)=K best individuals from E(t)∪C(t)
    end
    With the N best individuals from C(t)
    Q(t+1)=Output of translate operation on Q(t)
    Q(t+1)=Output of resize operation on Q(t+1)
    T = t+1
   end
```

2.1.1. *Initialising the Quantum Population*

A *quantum chromosome*, which is observed to give a specific solution vector of real-numbers, is made up of several quantum genes. The number of genes is determined by the required dimensionality of the solution vector. At the start of the algorithm, each quantum gene is initialized by randomly selecting a value from within the range of allowable values for that dimension. A gene's width value is set to the range of allowable values for the dimension. For example, if the known allowable values for dimension *j* are [-75, 75] then q_{ij} (dimension *j* in quantum chromosome *i*) is initially determined by randomly selecting a value from this range (say) -50. The corresponding width value will be 150. Hence, $q_{ij} = (-50,150)$. The square pulse need not be entirely within the allowable range for a dimension when it is initially created as the algorithm will automatically adjust for this as it executes. The height of the pulse arising from a gene *j* in chromosome *i* is calculated using

$$h_{ij} = \frac{1/\sigma_{ij}}{N} \tag{2}$$

where *N* is the number of individuals in the quantum population. This equation ensures that the probability density functions (PDFs) (see next subsection) used to generate the observed individual solution vectors will have a total area equal to one. Fig. 2 provides an illustration of a quantum gene where N=4.

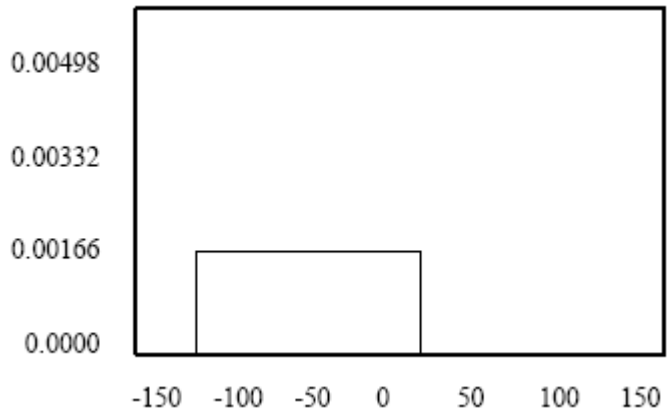


Fig. 2. A square pulse, representing a quantum gene, with a width of 150, centered on -50. The height of the pulse is 0.00166

2.1.2. *Observing the Quantum Chromosomes*

In order to generate a population of real-valued solution vectors, a series of observations must be undertaken using the population of quantum chromosomes (individuals). A pseudo-interference process between the quantum individuals is simulated by summing up the square pulses for each individual gene across all members of the quantum population. This generates a separate PDF (just the sum of the square pulses) for each gene and equation 2 ensures that the area under this PDF is one. Hence, the PDF for gene *j* on iteration *t* is

$$PDF_j(x) = \sum_{i}^{j} g_{ij} \tag{3}$$

where g_{ij} is the square pulse of the j^{th} gene of the i^{th} quantum individual (of *N*). To use this information to obtain an observation, the PDF is first converted into its corresponding Cumulative Distribution Function (CDF)

$$CDF_j(x) = \int_{L_j}^{U_j} PDF_j(x)dx \tag{4}$$

where U_j and L_j are the upper and lower limits of the probability distribution. By generating a random number *r* from (0,1) following a specific distribution, the CDF can be used to obtain an

observation of a real number *x*, where $x = CDF^{-1}(r)$. If the generated value *x* is outside the allowable real valued range for that dimension, the generated value is limited to its allowable boundary value. A separate PDF and CDF is calculated for each of the *G* gene positions. Once these have been calculated, the observation process is iterated to create a temporary population with *K* members, denoted *E(t)*.

2.1.3. Crossover Mechanism

The crossover operation takes place between *C(t)* and the temporary population *E(t)*. This step could be operationalized in a variety of ways with [5] choosing to adopt a variant of uniform crossover, without an explicit selection operator. After the *K* crossover operations have been performed, with the resulting children being copied into *E(t)*, the best *K* individuals $\in C(t) \bigcup E(t)$ are copied into *C(t)*.

2.1.4. Updating the Quantum Chromosomes

The next step is to update the corresponding width value of the j^{th} gene. The objective of this process is to vary the exploration / exploitation characteristics of the search algorithm, depending on the feedback from previous iterations. If the search process is continuing to uncover many new better solutions, then the exploration phase should be continued by keeping the widths relatively broad. However, if the search process is not uncovering many new better solutions, the widths are reduced in order to encourage finer-grained search around already discovered good regions of the solution space. There are multiple ways this general approach could be operationalized.

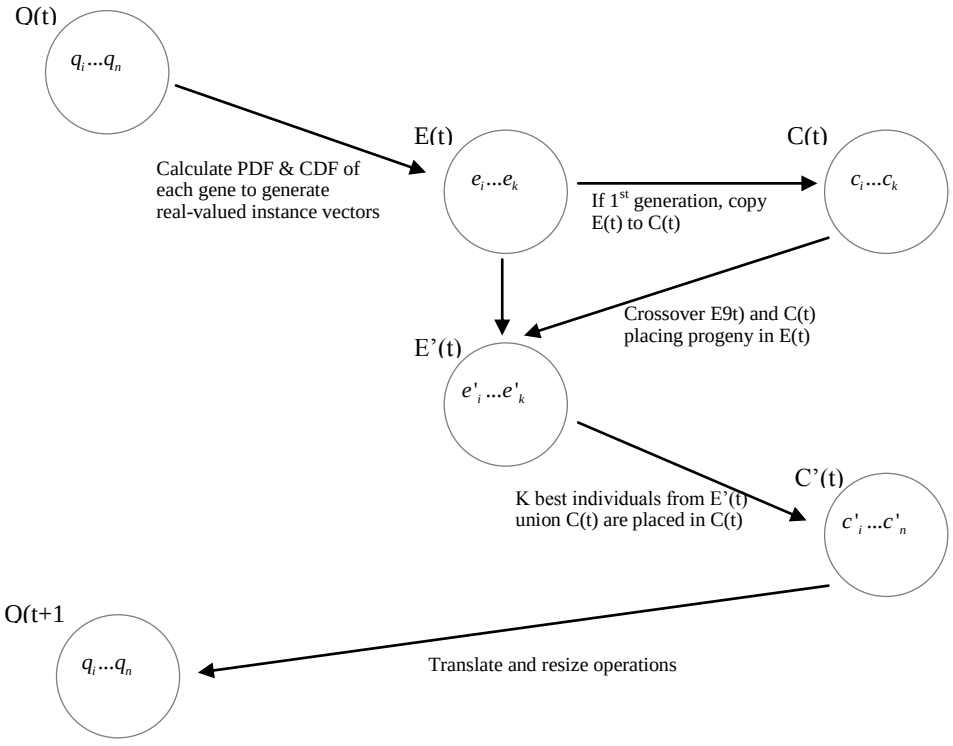


Fig. 3. Process in the creation of generation *t+1* from *t*

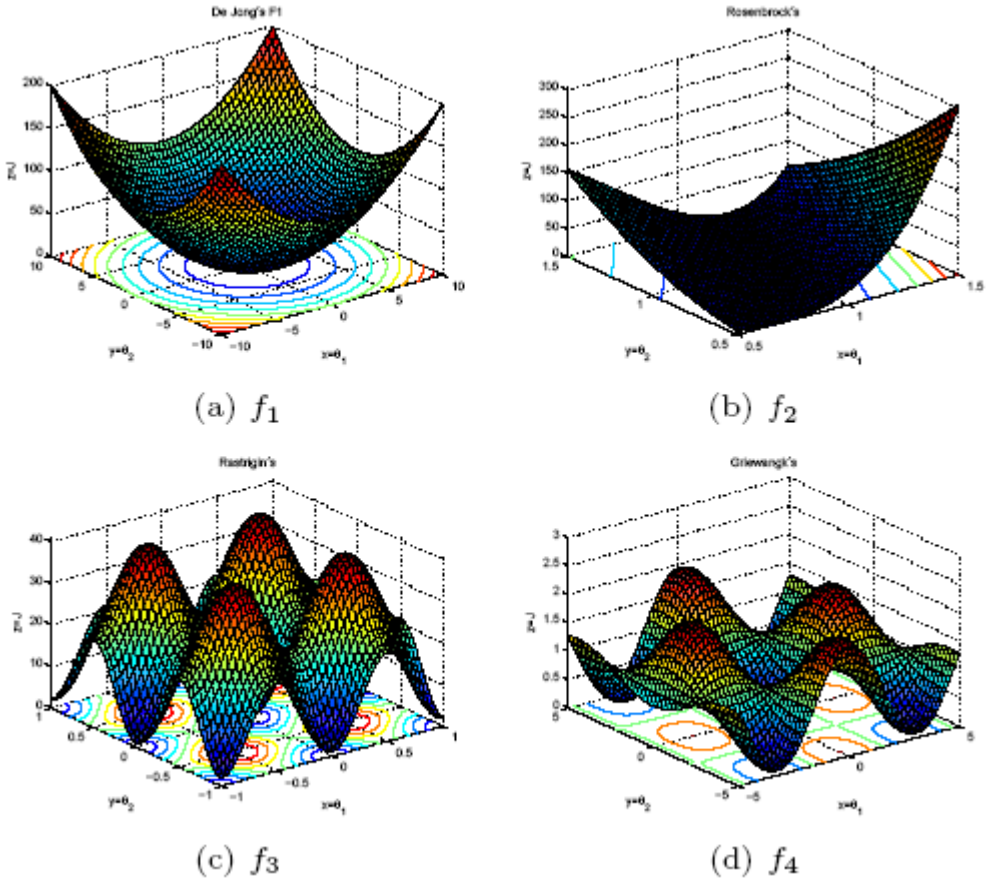
In this study we adjust the quantum probability amplitude by comparing each successive generation's best fitness so that the quantum chromosome can produce more promising individuals with higher probability in the next generation, i.e. if the best fitness has improved (disimproved) we shrink (enlarge) the width in order to improve the local (global) search. Figure 3 illustrates the process in the creation of generation from *t* to *t+1*.

2.2. *QIEA vs Genetic Algorithm*

A number of distinctions between the QIEA above and the GA can be noted. In the GA, the population of solutions persists from generation to generation, albeit in a changing form. In contrast, in QIEA, the population of solutions in *P(t)* are discarded at the end of each loop. The described QIEA, unlike GA, does not have explicit concepts of crossover or mutation. However, the adaptation of the quantum chromosomes in each iteration does embed implicit selection as the best solution is selected and is used to adapt the quantum chromosome(s). The crossover and mutation steps are also implicitly present, as the adaptation of the quantum chromosome in effect creates diversity, as it makes different states of the system more or less likely over time. Another distinction between the QIEA and the GA is that the GA operates directly on representations of the solution (the members of the current population of solutions), whereas in QIEA the update step is performed on the probability amplitudes of the ground states for each qubit making up the quantum chromosome(s).

3. Experimental Design

Four well-known static benchmark functions drawn from the evolutionary computation literature are chosen to test the ability of QIEA to find the global minimum within the search domain. We also test the scalability of the QIEA by varying the dimensionality of these benchmark functions, ranging from 100 to 1000 dimensions. The results are compared to those from canonical GA (CGA). Details of the benchmark functions are shown in Fig 4 and Table I.



(a) f_1 (b) f_2

(c) f_3 (d) f_4

Fig. 4. Two dimensional visualization of benchmark functions.

Table I Benchmark functions

f	Func. Name	Mathematical representation	Range	$f(x_i^*)$	x_i^*
f_1	DeJong	$f(x)=\sum_{i=1}^{p} x_i^2$	[-100,100]	0	0
f_2	Rosenbrock	$f(x)=\sum_{i=1}^{p-1} 100(x_{i+1}-x_i^2)^2+(1-x_i)^2$	[-100,100]	0	1
f_3	Rastrigrin	$f(x)=10p+\sum_{i=1}^{p}(x_i^2-10\cos(2\pi x_i))$	[-100,100]	0	0
f_4	Griewank	$f(x)=\sum_{i=1}^{p}\frac{x_i^2}{4000}-\prod_{i=1}^{p}\cos(\frac{x_i}{\sqrt{i}})+1$	[-100,100]	0	0

f_1 : DeJong's (Sphere) function is the simplest test function, being continuous, convex and unimodal.

f_2 : Rosenbrock's function is a classic optimization problem. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used to assess the performance of optimization algorithms.

f_3 : Rastrigrin's function is based on function 1 with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal if it has two or more local optima. However, the location of the minima are regularly distributed.

f_4 : Griewank's function is similar to Rastrigin's function. It has many widespread local minima. However, the location of the minima are regularly distributed.

3.1. *Static test*

The results from the QIEA are benchmarked against those from a binary-valued GA (*SGA*) and against results from a real-valued GA (*GEAT*). As the concept of a population varies between the GA and QIEA, in order to make a fair comparison between the methods, we allow each algorithm to perform a fixed number of fitness function evaluations (10,000 in all cases). This aims to give each algorithm the same chance to get fitness feedback from the environment. However of course, allowing each algorithm the same number of fitness evaluations does not ensure that each will have the same running time. As will be observed in the results, whilst using the same fitness function evaluation budget, the QIEA runs considerably faster than either version of the GA, demonstrating its run-time efficiency. The parameters used for QIEA and GA, selected from sensitivity test, are shown in Table II.

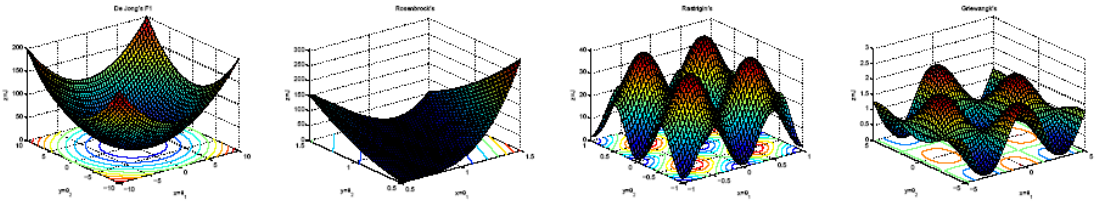
Table II. Optimal Parameter setting in GA and QIEA

SGA and GEAT	
Generation number	200
Population size	50
Mutation rate	0.005
Crossover rate	0.7
QIEA	
Generation number	1000
Population size	10
Crossover rate	0.1
Shrinkage factor	0.48
Enlargement factor	4.5
Resize base	(search bound) / 30

3.2. *Dynamic test*

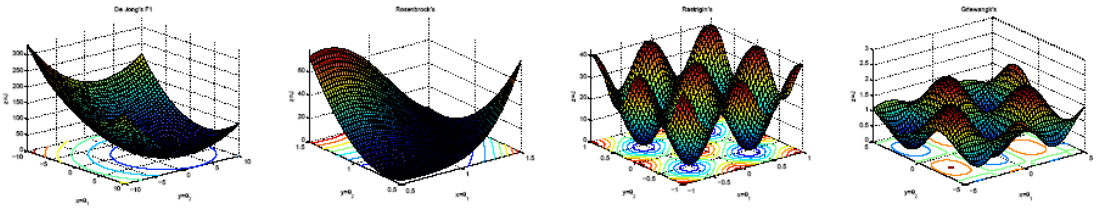
In the dynamic test, we employ the previous benchmark functions and move their central point with respect to the global minima randomly every 10 seconds. Fig. 5 and Fig. 6 depict the changing of the environment. For example, DeJong's function is set to standard form at t_0, as shown in Fig. 5(a). After 10 seconds, the global minima moves along the axis randomly, and the new optima changes from [0, 0] to [2.5, 5] respectively, as shown in Fig. 6(a).

In this dynamic environment, the object is to find the optimum, and then track it successfully over time during the testing period. The overall testing period is 1 minute, i.e. 6 optimizations for each algorithm.



(a) f_1 (b) f_2 (c) f_3 (d) f_4

Fig 5 Two dimensional visualization of benchmark functions at start time



(a) f_1 (b) f_2 (c) f_3 (d) f_4

Fig 6 Two dimensional visualization of benchmark functions after 10 seconds

The real-valued quantum-inspired evolutionary algorithm and binary genetic algorithm are examined in these experiments. As the environment alters every 10 seconds, the algorithm parameters are adjusted to suit this dynamic environment and are shown in table III.

Table III. Optimal Parameter setting in GA and QIEA for dynamic test

SGA	
Generation number	10
Population size	5
Mutation rate	0.005
Crossover rate	0.7
QIEA	
Generation number	10
Population size	10
Crossover rate	0.1
Shrinkage factor	0.2
Enlargement factor	5
Resize base	(search bound) / 30

4. Results

4.1. *Static test*

The results of static test are shown in Tables IV to VII. In all cases, the results are averaged over 30 runs of each algorithm. The first column lists the minimal (optimal) objective value found during the 30 runs within the whole population. The second and third column lists the mean and standard deviation for the minimal value of 30 runs. The *Time* column shows the average time to complete each of the 30 runs, measured in seconds.

Table IV. f_1 (DeJong) results

Algorithm	Best	Mean	S.D	Time(s)
Dimension: 100				
SGA	7836	9261	1044	6
GEAT	121703	122161	1697	6
QIEA	5663	7268	980.1	2
Dimension: 500				
SGA	6.01e+5	6.21e+5	1.12e+4	30
GEAT	1.51e+6	1.54e+6	2.90e+4	12
QIEA	5.73e+5	6.25e+5	2.21e+4	3
Dimension: 1000				
SGA	1.73e+6	1.81e+6	3.15e+4	77
GEAT	3.08e+6	3.13e+6	6.79e+4	22
QIEA	1.73e+6	1.81e+6	3.72e+4	4

Table V. f_2 (Rosenbrock) results

Algorithm	Best	Mean	S.D	Time(s)
Dimension: 100				
SGA	1.64e+9	2.41e+9	4.51e+8	7
GEAT	2.19e+10	3.28e+10	4.60e+8	6
QIEA	2.38e+8	4.18e+8	9.67e+7	2
Dimension: 500				
SGA	2.08e+11	2.31e+11	1.35e+10	36
GEAT	8.44e+11	8.76e+11	2.04e+10	12
QIEA	1.96e+11	2.25e+11	1.38e+10	3
Dimension: 1000				
SGA	7.42e+11	7.71e+11	2.05e+10	77
GEAT	1.75e+12	1.82e+12	3.66e+10	22
QIEA	7.56e+11	8.41e+11	3.70e+10	4

Table VI. f_3 (Rastrigrin) results

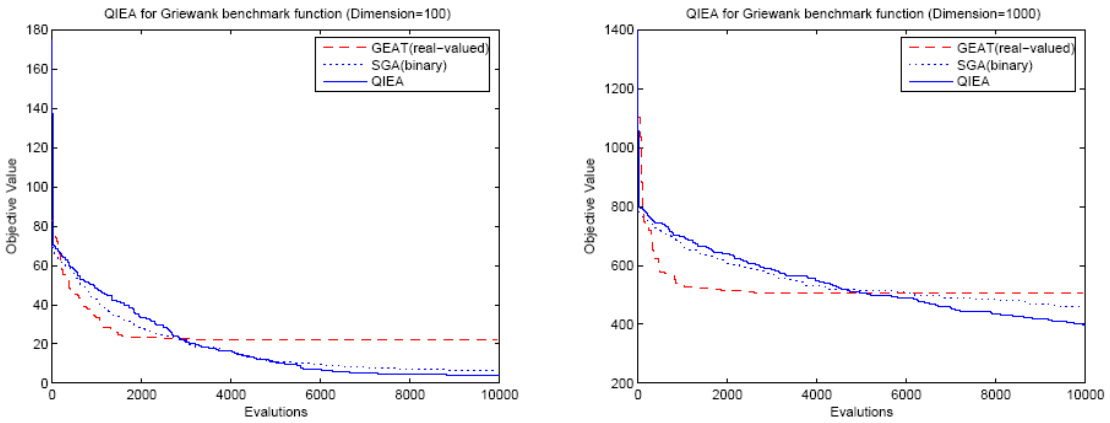
Algorithm	Best	Mean	S.D	Time(s)
Dimension: 100				
SGA	19250	24242	2546	7
GEAT	119547	161492	19758	6
QIEA	7285	9585	1305	2
Dimension: 500				
SGA	6.18e+5	6.62e+5	2.24e+4	56
GEAT	1.43e+6	1.52e+6	1.49e+5	12
QIEA	5.69e+5	6.43e+5	2.51e+4	3
Dimension: 1000				
SGA	1.77e+6	1.90e+6	5.33e+4	126
GEAT	3.15e+6	3.16e+6	6.97e+4	22
QIEA	1.70e+6	1.82e+6	4.76e+4	5

Table VII. f_4 (Griewank) results

Algorithm	Best	Mean	S.D	Time(s)
Dimension: 100				
SGA	4.27	5.54	0.50	7
GEAT	32.10	37.72	1.88	6
QIEA	2.29	2.79	0.24	3
Dimension: 500				
SGA	157.00	167.95	6.73	56
GEAT	339.54	384.75	8.85	13
QIEA	130.58	155.80	6.51	4
Dimension: 1000				
SGA	441.58	463.92	9.42	137
GEAT	730.80	781.43	20.69	24
QIEA	440.59	462.89	9.39	5

Fig. 7(a) and 7(b) illustrate the performance of QIEA against that of the binary and real-valued GA. GEAT converges most quickly, but after approximate 3000 to 5000 evaluations, QIEA generally outperforms both types of GA. Across all the twelve experiments (four functions times three levels of dimensionality), we can see that QIEA achieves or equals the best result in eight experiments. The results also indicate that the QIEA performs well as problem dimensionality increases, with the QIEA beating or equaling the best GA result on three of the four 1000 dimension experiments.

Looking at the runtime metrics, it also becomes clear that the the above results are actually a conservative assessment of the QIEA's performance. Fig. 8 graphs the runtime for each method on each problem instance, across the three levels of dimensionality. This allows assessment of the runtime scalability of each algorithm. It is observed that the QIEA has a much lower run time than either of the two GA variants with the latter having run times ranging from three to eighteen times those of the QIEA.



(a) Global Search within 100 dimensions (b) Global Search within 1000 dimensions

Fig. 7 Global Search from 100 dimensions to 1000 dimensions

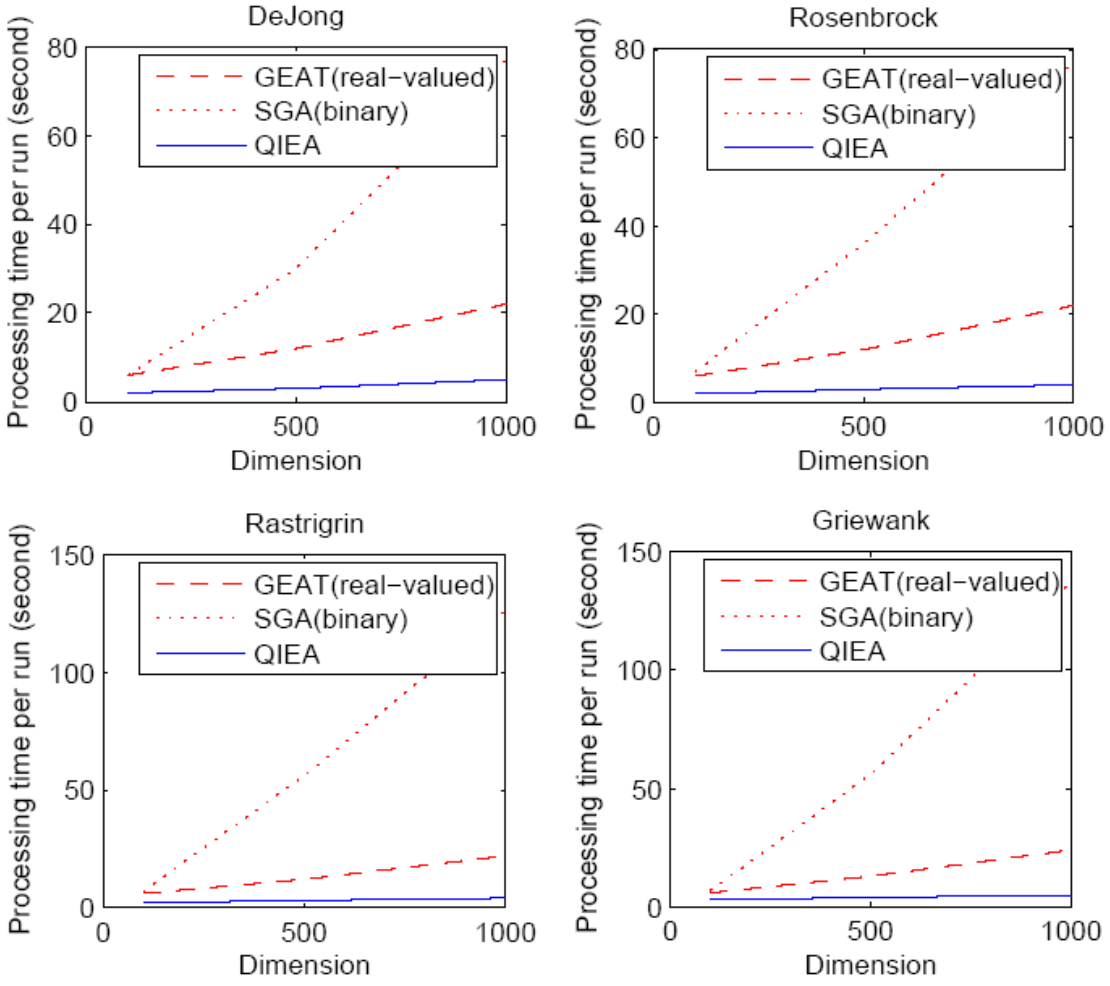


Fig. 8 Processing time for benchmark functions

An alternative way of comparing the algorithms relative performance would be to allow them a fixed run-time budget. The results obtained suggest that this would favor the QIEA. In summary, the results provide support for a claim that the real-valued QIEA could have substantial utility for high dimensional real-world problems.

4.2. *Dynamic test*

The results of dynamic test are shown in Tables VIII to XI. Similar to static test, the results are averaged over 30 runs of each algorithm. The first column lists the minimal (optimal) objective value found during the 30 runs within the whole population. The second and third column lists the mean and standard deviation for the minimal value of 30 runs.

Table VIII. f_1(DeJong) results

Algorithm	Best	Mean	S.D
Dimension: 100			
SGA	1.74e+5	1.92e+5	8.34e+3
QIEA	1.87e+5	2.11e+5	1.12e+4
Dimension: 500			
SGA	1.31e+6	1.35e+6	2.28e+4
QIEA	1.45e+6	1.49e+6	2.11e+4
Dimension: 1000			
SGA	2.84e+6	2.89e+6	1.94e+4
QIEA	2.71e+6	2.75e+6	1.01e+4

Table IX. f_2(Rosenbrock) results

Algorithm	Best	Mean	S.D
Dimension: 100			
SGA	7.12e+10	8.68e+10	6.25e+9
QIEA	7.63e+10	1.02e+11	1.02e+10
Dimension: 500			
SGA	6.84e+11	7.27e+11	2.12e+10
QIEA	6.00e+11	7.16e+11	2.55e+10
Dimension: 1000			
SGA	1.57e+12	1.64e+12	2.80e+10
QIEA	1.47e+12	1.55e+12	1.23e+10

Table X. f_3(Rastrigrin) results

Algorithm	Best	Mean	S.D
Dimension: 100			
SGA	1.71e+5	1.96e+5	9.34e+3
QIEA	2.01e+5	2.20e+5	1.08e+4
Dimension: 500			
SGA	1.32e+6	1.36e+6	2.30e+4
QIEA	1.30e+6	1.36e+6	2.21e+4
Dimension: 1000			
SGA	2.82e+6	2.91e+6	3.68e+4
QIEA	2.70e+6	2.76e+6	1.04e+4

Table XI. f_4(Griewank) results

Algorithm	Best	Mean	S.D
Dimension: 100			
SGA	46.97	49.87	1.91
QIEA	53.19	60.56	3.11
Dimension: 500			
SGA	332.8	341.0	5.40
QIEA	320.6	337.2	7.04
Dimension: 1000			
SGA	703.4	727.3	8.7
QIEA	687.7	695.1	1.4

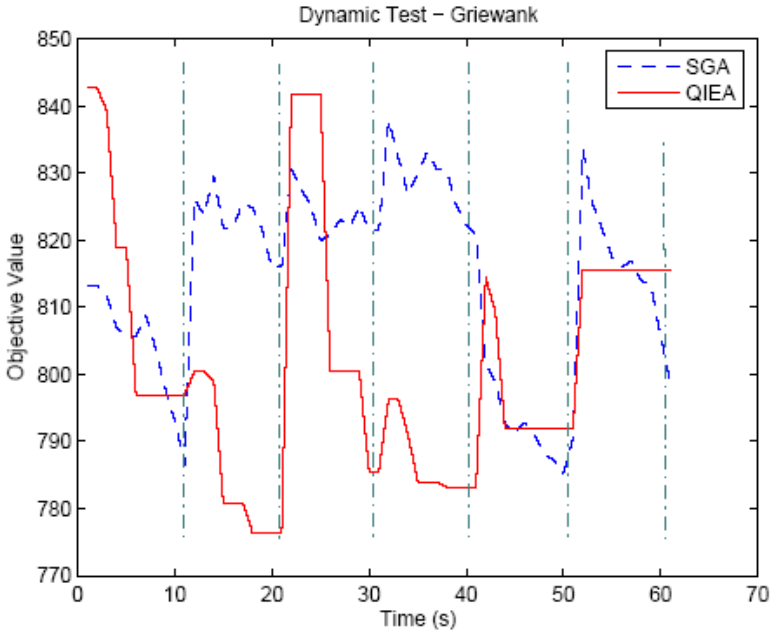


Fig. 9 Global Search within 1000 dimensions

The results show that although QIEA does not work better than SGA within low dimensions, but it outperforms SGA within high dimensions and complicated benchmark functions, in accordance with the static test. Fig. 9 illustrates the performance of QIEA against that of the canonical GA for Griewank benchmark function. As introduced in Section 3.2, the time interval is 10 seconds; hence it is a high-frequency moving environment. At the end of each time window, i.e. at time point 10, 20, 30, 40, 50 and 60, both QIEA and SGA stop their evolution and start another optimization as a new search. The results show that SGA can find better results at time 10, 50 and 60, and QIEA outperforms SGA at time 20, 30 and 40. The optimization of QIEA is more stable than SGA, i.e. the standard deviation of QIEA is smaller than SGA. The parameters of SGA, such as population size and generation number, have to be adjusted to suit the time requirement, hence the algorithm cannot work as well as in static environment, while quantum genes in QIEA can still work well. This advantage of QIEA can be applied for solving real-world problems, such as algorithmic trading. As the financial data changes very quickly the processing time of a trading algorithm is critical (see [2; 3; 4] for an illustration of current work applying other natural computing algorithms in finance). The QIEA will be examined for those applications in future work. We also notice that the SGA evolves more gradually than QIEA over generations within each time window. That difference will be further investigated.

4.3. *Sensitivity Analysis*

In order to gain greater insight into the operation of the algorithm, and to guide future applications of it, we undertook an analysis by systematically investigating a variety of parameter settings for shrinkage and enlargement. The results of the optimal objective value as a function of the enlargement and shrinkage parameters are reported in Table XII. The crossover rate is fixed at 0.1, a population size of 10 and a generation number of 10 are used. Fig. 10 graphs these results. Respect to the global minima (63.17), the optimal enlargement factor is 5, and the shrinkage factor is 0.2, hence these parameters were used for our dynamic environment tests.

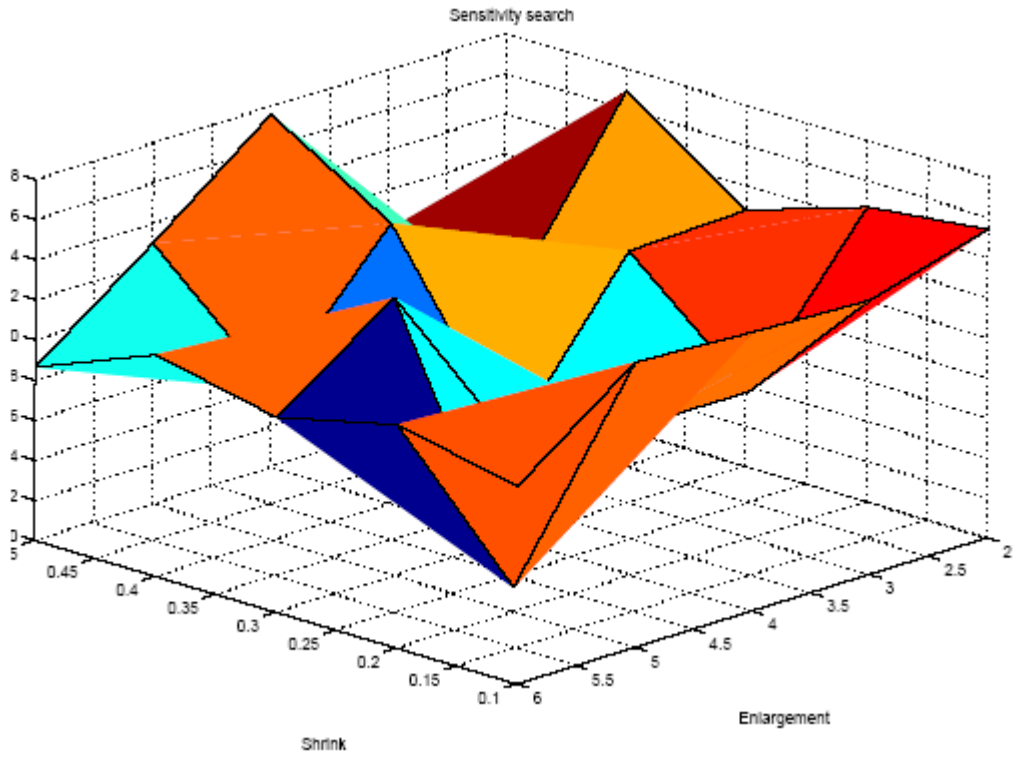


Fig. 10 Sensitivity search for QIEA benchmark test

Table XII. Optimal Parameter setting in GA and QIEA

Enlarge\Shrink	0.1	0.2	0.3	0.4	0.5
2	75.63	73.20	70.45	76.50	67.35
3	73.71	67.53	73.74	67.34	71.59
4	71.54	68.06	66.75	71.56	77.54
5	77.39	72.20	63.17	68.57	73.71
6	67.56	70.87	69.51	70.16	67.85

5. Conclusions

Following earlier claims that QIEAs may offer advantages in high dimensional environments, this study tests a real-valued QIEA on a series of benchmark functions of varying dimensionality in order to examine its scalability and efficiency. The results are compared with those from a canonical genetic algorithm. The comparison shows that the QIEA obtains highly competitive results versus the genetic algorithm in static tests, while outperforming the canonical GA in terms of stability in dynamic tests. This suggests that QIEA may have substantial utility for real-world high dimensional and dynamic problems.

A particularly interesting avenue of study would be to examine the utility of the QIEA in financial trading environments where price / volume data is being generated multiple times per second. For any real-time trading systems and pricing models, processing time and efficiency are crucial, especially in financial derivatives markets. Future work will target these high-frequency and challenging problems.

Furthermore, in this paper, the only distribution employed by QIEA is the normal distribution. Future work will explore the utility of other distributions to explore their impact on the chromosome evolution process.

References

1. Baluja, S. (1994). Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. *Technical Report*, No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania.
2. Brabazon, A. and O'Neill, M. (2006). *Biologically Inspired Algorithms for Financial Modelling*, Springer: Berlin.
3. Brabazon, A. and O'Neill, M. (eds) (2008). *Natural Computing in Computational Finance*, Springer: Berlin.
4. Brabazon, A. and O'Neill, M. (eds) (2009). *Computational Intelligence in Finance*, Springer: Berlin (forthcoming).
5. da Cruz, A., Vellasco, M. and Pacheco, M. (2006). Quantum-inspired evolutionary algorithm for numerical optimization, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, 16-21 July, Vancouver, pp. 9180--9187, IEEE Press.
6. Fan, K., Brabazon, A., O'Sullivan, C. and O'Neill, M. (2007). Option Pricing Model Calibration using a Real-valued Quantum-inspired Evolutionary Algorithm, *Genetic and Evolutionary Computation Conference (GECCO) 2007*, 1983-1990, Springer London, ACM Press.
7. Fan, K., Brabazon, A., O'Sullivan, C. and O'Neill, M. (2008). Benchmarking the performance of the real-valued quantum-inspired evolutionary algorithm, *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, 1-6 June, Hong Kong, pp.3092-3098.
8. Fan, K., O'Sullivan, C., Brabazon, A. and O'Neill, M. (2008). Testing a Quantum-inspired Evolutionary Algorithm by Applying It to Non-linear Principal Component Analysis of the Implied Volatility Smile, *Natural Computing in Computational Finance*, Springer, pp. 89-108
9. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley.
10. Han, K-H. and Kim, J-H. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE Transactions on Evolutionary Computation*, 6(6):580--593.
11. Han, K-H. and Kim, J-H. (2002). Quantum-inspired evolutionary algorithms with a new termination criterion, Hεgate and two-phase scheme, *IEEE Transactions on Evolutionary Computation*, 8(3):156--169.
12. Harik, G., Lobo, F. and Goldberg, D. (1998). The compact genetic algorithm, *Proceedings of the International Conference on Evolutionary Computation (CEC 1998)*, pp. 523-528, New Jersey: IEEE Press.

13. Jeremy, S. De Bonet, et al(1997): MIMIC: Finding Optima by Estimating Probability Densities, *Advances in Neural Information Processing System*, 1997, Vol. 9, pp. 424-431.
14. Khan, N., Goldberg, D. and Pelikan, M. (2002). Multiobjective bayesian optimization algorithm, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, pp. 648, Morgan Kaufmann: San Mateo.
15. Larranaga, P. Lozano, J. (Eds.) (2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer.
16. Muhlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5 (3), 303--346.
17. Muhlenbein,H. and Mahnig,T.(1999). The Factorized Distribution Algorithm for additively decomposed functions. *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999)*, IEEE press,752-759.
18. Narayanan, A. and Moore, M. (1996). Quantum-inspired genetic algorithms, *Proceedings of IEEE International Conference on Evolutionary Computation (CEC1996)*, May 1996, pp. 61--66, IEEE Press.
19. Pelikan, M., Goldberg,D.E. and Cantú-paz, E.(2000). LinkageProblem, Distribution Estimation and Bayesian Networks. *Evolutionary Computation*, 8(3):311-340.
20. Pelikan, M. (2005). *Hierarchical Bayesian Optimization Algorithm Toward a New Generation of Evolutionary Algorithms*, Springer: Berlin.
21. Pelikan, M., Sastry, K., Cantu-Paz, E. (Eds.). (2006). *Scalable Optimization via Probabilistic Modeling*, From Algorithms to Applications Series, Springer: Berlin.
22. Pohlheim, H. (2007). Geatbx: Genetic and Evolutionary Algorithm Toolbox for Use With Matlab. http://www.geatbx.com
23. Tang, K., Yao, X., et al, (2007). Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization, *Technical Report*, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php
24. Thierens, D. and Bosman, P. (2001). Multi-objective mixture-based iterated density estimation evolutionary algorithms, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, San Francisco, pp. 663-670, Morgan Kaufmann: San Mateo.
25. Yang, S., Wang, M. and Jiao, L. (2004). A genetic algorithm based on quantum chromosome, *Proceedings of IEEE International Conference on Signal Processing (ICSP 04)*, 31 Aug- 4 Sept. 2004, pp. 1622--1625, IEEE Press.
26. Yang, S., Wang, M. and Jiao, L. (2004). A novel quantum evolutionary algorithm and its application, *Proceedings of IEEE Congress on Evolutionary Computation 2004 (CEC 2004)*, 19-23 June 2004, pp. 820--826, IEEE Press.
27. Yang, S. (2005). Memory-enhanced Univariate Marginal Distribution Algorithms for Dynamic Optimization Problems, *Proceedings of Congress on Evolutionary Computation (CEC 2005)*, pp. 2560-2567, IEEE Press.
28. Zhou, S. and Sun, Z.(2005): A New Approach Belonging to EDAs: Quantum-Inspired Genetic Algorithm with Only One Chromosome. *ICNC 2005, LNCS 3612*: pp. 141-150.

About the authors

Kai Fan is a PhD student in finance in the School of Business at University College Dublin. He did semiconductor physics before and is working on financial modelling in the Natural Computing Research and Applications Group at UCD (http://ncra.ucd.ie). He has extensive financial consulting experience and his research interests include financial derivatives pricing and algorithmic trading. E-mail: kai.fan@ucd.ie

Prof. Brabazon is currently Head of Research in the School of Business at University College Dublin. He is also co-founder and co-director, of the Natural Computing Research and Applications Group at UCD (http://ncra.ucd.ie). His primary research interests concern the development of natural computing theory and the application of natural computing algorithms in finance. He is a member of the IEEE Computational Finance and Economics Technical Committee.
E-mail: anthony.brabazon@ucd.ie

Dr. O'Sullivan is a lecturer in finance in the School of Business at University College Dublin. His primary research interests are in numerical methods for derivatives pricing, financial modelling and the application of natural computing algorithms in finance. He is the director of the M.Sc. in Quantitative Finance at UCD.
E-mail: conall.osullivan@ucd.ie

Dr. O'Neill is a faculty member of the School of Computer Science and Informatics at University College Dublin. He is co-founder and co-director of UCD's Natural Computing Research &

Applications group (http://ncra.ucd.ie), and has authored a number of books including *Grammatical Evolution* and *Biologically Inspired Algorithms for Financial Modelling*. He is co-founder and co-Chair of EvoFIN, the European Workshop on the application of Evolutionary Computation in Finance and Economics.

E-mail: m.oneill@ucd.ie