Convergence Properties of Quantum Evolutionary Algorithms on High Dimension Problems

Joe Wright*, Ivan Jordanov
School of Computing, University of Portsmouth, Lion Terrace, Portsmouth, PO1 3HE, UK; emails: Jonathan.Wright@port.ac.uk; Ivan.Jordanov@port.ac.uk

Abstract. We propose and investigate new rotation gates for two modified Quantum Inspired Evolutionary methods for solving high dimension optimisation problems. The *Quantum Inspired Evolutionary Algorithms* (*QIEA*) were originally used for solving binary encoded problems and their signature features follow superposition of multiple states on a quantum bit and a rotation gate. In order to apply this paradigm to high dimension problems, we propose two quantum methods *Half Significant Bit* (*HSB)* and *Stepwise Real QEA (SRQEA*), developed using binary and real encoding respectively, while keeping close to the original quantum computing metaphor. We introduce five performance metrics and use them to evaluate the proposed approaches against sets of multimodal mathematical test functions and real world problems of high dimensionality. We report issues found while implementing some of the published real *QIEA* techniques which were the motivation for developing our real algorithm modifications. Our methods focus on introducing and implementing new rotation gate operators used for evolution, including a novel mechanism for preventing premature convergence in the binary algorithm. The applied performance metrics show superior results for our quantum methods on most of the test problems (particularly with high dimension problems), demonstrating faster convergence and accuracy.

Keywords: quantum evolutionary methods, estimation of distribution algorithms, performance metrics, multimodal functions, high dimension problems, global optimization.

1. Introduction

A challenge for modern computer science is the development of algorithms for increasingly complex optimisation problems. These may include a variety of practical real-world problems, such as structural engineering [1,2], 3D mesh simplification [3], antenna design [4], wireless network design [5], electric power systems [6], resource allocation [7], digital image watermarking [8], EEG classification [9], benchmark problems [10], large data set analysis [11], or mathematical functions designed to test or challenge aspects of optimisation [12,13]. Approaches to solving these problems include typical algorithms such as particle swarm optimisation (PSO) [14,15], genetic algorithms (GA) [16,17], differential evolution [18–20], and other nature inspired methods, e.g., honey bee [21] and cloud drops algorithms [22].

As the number of dimensions increases, the optimisation task becomes more difficult as a larger solution space must be searched, which in turn increases computational demands. One approach to deal with these demands is to pre-filter the data in some way, in order to reduce the number of dimensions [23,24]. Additionally, transforming the data so that interpretation becomes simpler, can allow easier classification and optimization, with a typical example being the use of sparse representation to obtain a linear problem [25]. Examples of applications of these techniques include feature extraction in computer vision [26], face recognition [27] or related image analysis [28].

In 2002 a new optimization algorithm was presented in [29], that took inspiration from quantum computing to evolve a probability distribution, which in turn was employed to search a solution space. The method used a string of quantum bits (Qbit), each storing sampling probability of a one or a zero. Successive sampling of the string produced a series of candidate binary solutions. If any of these were found to be an improvement, the underlying Qbit probabilities are adjusted to make the candidate more likely to appear in successive samples. A detailed explanation of the algorithm is presented in section 2.

Originally, this quantum-inspired evolutionary algorithm (*QIEA*) was applied to the *Knapsack problem* - a binary combinatorial optimisation problem [29], and then modified versions were applied by others to *OneMax*, *Noisy-flat* and *NK-landscapes* [30], neural-network training [31], and networking [32].

Although some attempts have been made to apply binary *QIEA* to real-value problems [33], most applications to such tasks have used real-value *QIEA* [34–38]. These algorithms took, at least superficially, the concepts of superposition and quantum rotation gates that were introduced with the binary *QIEA*, and adopted them for application to real-value problems. However, when reviewing them we encountered a number of problems. Many were incompletely described and could therefore not be reproduced, one was trivial to implement [34] but performed extremely poorly on a set of multimodal mathematical test functions, and of greatest concern, one paper [35] claimed superior performance to another optimization algorithm that was later found to not have performed as well as claimed

[39]. A second issue, more of a philosophical concern than a practical problem, is that in making the adaptation to real-value problems, the purity of the original quantum inspiration (that are naturally applied to binary problems) may be lost. We discuss these concerns in sections 3 and 6. Various attempts at a real *QIEA* can be found in the literature, including [36–38], and in [40] a review is presented of both binary and real *QIEA*. In this investigation we have chosen [41] to build a real-coded *QIEA* upon, as it performed the best in initial tests and contained features common to many real *QIEA*.

The goals of the research presented here were to see how the *Classic* version [29] of the binary *QIEA*, as well as a representative real *QIEA*, would perform on a number of recent benchmark test functions and several real-world problems, and to investigate, design, and develop modified binary and real *QIEA*, proposed to improve the performance of these approaches in terms of convergence and accuracy.
In sections 2 and 3 we present the binary and the real *QIEA* under investigation, including our modifications. Section 4 outlines the methods used for testing, the obtained results are presented and discussed in section 5, and the conclusion of this paper is given in section 6.

2. Binary *QIEA*

This section presents the original binary quantum inspired evolutionary algorithm (*bQIEA*) [29], along with a preliminary investigation highlighting arising problems when applying it to real-value tasks. We then introduce a modified method designed to tackle these issues.

2.1. Classic QIEA

The original *QIEA* [29], hereon in labelled *Classic*, contains the core properties of *QIEA*: *Qbit* sampling; and the rotation gate operator. Unlike a traditional binary evolutionary algorithm, *Classic* stores a string of probability values called Qbits. For each individual i of length N bits in a population of size p, a pair of values for bit j gives the probability of sampling a zero or a one at iterations t, shown with quantum state notation in Eq. (1). Through repeated iterations of sampling, the same Qbit value can be used to sample a sequence of random binary values. We can also interpret the pair of probabilities as an angle (Eq. (2)), and when this angle has a value of $\pi/4$ (highest entropy), both one and zero have an equal chance of being sampled. An angle near $\pi/2$ favours sampling 1s, and a value close to zero favours sampling 0s.

Even in the absence of evolution of the chromosomes, *Classic* will continue to produce different candidates for the fitness function, unlike a traditional evolutionary algorithm. The combination of probability and sampling is inspired by the quantum computing principal of superposition. Superposition is the ability of a Qbit to hold multiple states simultaneously. The string Q_i therefore provides a probability distribution function for generating candidate solutions C_i at each iteration.

While random sampling allows the solution space to be searched, the Qbits need to be changed in order to localise and refine the search. Using the angle interpretation of the Qbit, an update modifier called a rotation gate can be used, which simply shifts the angle, and therefore the probability, one way or the other. By using the best solution found so far (called the attractor A_i) for an individual, this gate can be made to rotate towards a position that reinforces the attractor probabilities, if it is still the best solution, or away, if the candidate was better. The magnitude of rotation $|\Delta\theta|$ is fixed to $\pi/100$ and the Qbit is restricted within the range $(0, \pi/2)$. The rotation gate is given in Eq. (3).

Information is distributed around the population via the attractors A. Every *G-th* iteration, a global migration is performed, where the best attractor in the population is copied to all individuals and every *L-th* iteration, a local migration is conducted, where the best attractor in a subset of the population is copied to the whole subset. For the investigations presented here, $G=20$, $L=1$ (meaning improvements to attractors are copied to subsets at the end of each iteration), and the number of subset groups is assumed to be *5*. These values are adopted from [29], where they were established to be successful and we do not investigate them further. Subset allocation is done simply by splitting the full population into equally sized groups of individuals.

$$\begin{aligned} &\left|\Psi_j\right\rangle = \alpha_j\left|0\right\rangle + \beta_j\left|1\right\rangle \\ &P\left(\Psi_j = 1\right) = \left|\alpha_j\right|^2 \\ &P\left(\Psi_j = 0\right) = \left|\beta_j\right|^2 \\ &\left|\alpha_j\right|^2 + \left|\beta_j\right|^2 = 1 \end{aligned} \quad , \qquad (1)$$

$$\left|\alpha_j\right|^2 = \sin^2\left(Q_{i,j}\right),$$
$$0 \le Q_{i,\mathrm{j}} \le \frac{\pi}{2}. \tag{2}$$

$$Q_{\mathrm{i+1,j}} = \begin{cases} \min\left(Q_{\mathrm{i,j}} + \Delta\theta, \dfrac{\pi}{2}\right), & \Delta\theta > 0 \\ \max\left(Q_{\mathrm{i,j}} + \Delta\theta, 0\right), & \Delta\theta < 0 \end{cases}, \tag{3}$$

where *Δθ* is the size and direction of rotation.

2.2. Application to real-value problems and convergence issues

In our investigation, for the binary optimization algorithms, real values are encoded using a simple scheme. Binary strings of length 24 bits are used to generate numbers in the [0, 2^{24}-1] range, which are then linearly mapped onto the domain for the fitness function being optimized.

An initial application of *Classic* to real-valued problems highlighted a convergence issue. A plot of a typical evolution is shown in Fig. 1a, where the least significant Qbits (LSBs) are saturating before the most significant Qbits. Once a Qbit saturates, it will no longer evolve because sampling will continuously produce ones or zeros, depending on which end of the scale the Qbit has saturated to. This means that the LSBs had become randomly fixed relatively early on in the optimization, thus preventing fine scale exploitation.

For reasonably smooth search spaces, the early stages of the search should focus on finding the general locations of extrema, rather than refining solutions to a precise position. During this phase, the fitness function will be affected more by large movements than by small ones. With a binary representation, this will manifest in the most significant bits (MSBs) dominating the search, as changes to them are likely to find larger improvements to the fitness than changes to the LSBs.

Therefore, in the early stages, the LSBs provide little selection pressure, and so random values for these bits will be tolerated, while the MSBs are optimised. We can model this by assuming that the LSBs contribute nothing to the fitness evaluation, and so the LSBs of the best candidate will always be regarded as ‘better’ whether they sample a one or a zero. As the rotation gates are applied to adjust the Qbit probabilities to reinforce the sampled state, the LSBs (in the absence of exerting evolutionary pressure) will follow a simple, but non-symmetrical random walk, where the probability of rotating the Qbit probability towards an extremum (one or zero) increases as it moves away from the centre. This process is expressed with Eq. (3) and ten example simulations of the process are shown in Fig. 2, demonstrating quick convergence to either the one or zero limits.

$$\begin{cases} X_t < 0 & y_{t+1} = \max\left(0, y_t + X_t\Delta\theta\right) \\ X_t > 0 & y_{t+1} = \min\left(1, y_t + X_t\Delta\theta\right) \end{cases},$$
$$X_t \sim \begin{cases} -1, & p = 1 - y_t \\ 1, & p = y_t \end{cases}, \tag{4}$$

where X_t is a random variable with a Bernoulli distribution, with the probabilities for the two states being dependent on the random walk position y_t at time *t*. The step size for the rotation gate is *Δθ*.

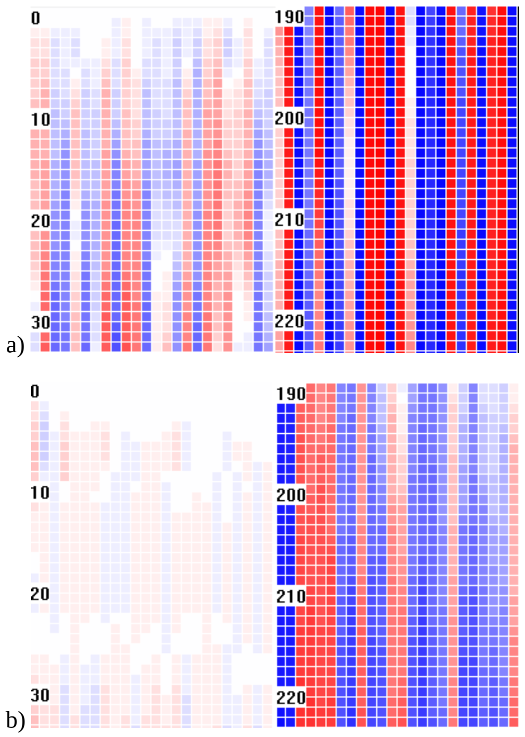


Fig. 1. Evolution of Qbit probabilities on *Griewank* function using (a) *Classic* and (b) *HSB* algorithms. Bits for one real value are shown with most significant bits to the left. Early in the evolution, all squares are pale (mid-range). Later on, for *Classic*, the LSBs (to the right) are all saturated, while several of the MSBs are paler and still undergoing evolution. For *HSB* however, limiting saturation of a Qbit to be no more than the current value of the neighbour with half bit index (more significant), prevents the LSBs from saturating before the MSBs.

In reality, the LSBs will exhibit some evolutionary pressure, varying according to the shape of the fitness landscape, but as illustrated in Fig. 1a, the time line of the Qbit evolution shows that the LSBs can be observed to saturate early on in the process.

2.3. Improved bQIEA convergence performance for real value problems – HSB (Half Significant Bit).

One possible solution of these convergence problems is presented in [33], where the rotation gate operator has limits imposed that were slightly within the zero to one range. This means that, even late in the evolution, it is always possible to sample new bit values as the Qbits never completely saturate.

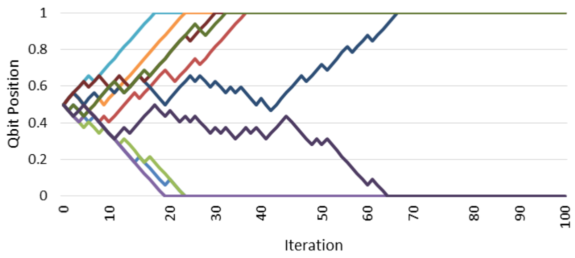


Fig. 2. Ten example simulations of LSB random walk process when they exert relatively little pressure on the evolution. Each colour represents a different simulation run, the vertical axis is Qbit position, with each run starting in the central 0.5 position, and the horizontal axis is the number of iterations. Runs' quick saturation to either one or zero is showing a tendency for the LSB to prematurely converge if they do not exert significant pressure on the evolution, using the standard QIEA rotation gate.

However, as we have analysed this premature convergence to be a problem of LSB evolution relative to MSB revolution, and inspired by early experimentation that failed to find much benefit from the constraint strategy, we present and test a method that explicitly constrains LSB Qbit rotation, relative to MSB Qbit rotation. When rotating a Qbit, we impose a limit upon the range that it can move to, based on the current value of a more significant bit, so that it cannot move to a more extreme value. This has the effect of delaying large movements in the LSBs until the MSBs have saturated.

Initially, we experimented with using the more significant immediate neighbour bit as a limiting condition, but found this to slow the convergence, so settled upon picking a bit index that was half the position value of the Qbit being rotated (assuming bit index zero as the most significant one). This is a somewhat less aggressive limiting condition, which gives a compromise between premature convergence and overly slow convergence. Future work will be needed to identify the optimum index strategy. The adjusted formula for the rotation is given in Eq. (5), with the general algorithm code staying the same as for *Classic*. This modified algorithm is called *HSB* (*Half Significant Bit*) in this paper, and preliminary results of an evolution are shown in Fig. 1b. The global and local migration rates G=20 and L=1, and the population subdivisions (5 subsets) are assumed the same as in the *Classic* method [29].

$$Q_{i+1,j} = \begin{cases} \min\left(Q_{i,j} + \Delta\theta, \pi/4 + \left|Q_{i,h} - \pi/4\right|\right), & \Delta\theta > 0 \\ \max\left(Q_{i,j} + \Delta\theta, \pi/4 - \left|Q_{i,h} - \pi/4\right|\right), & \Delta\theta < 0 \end{cases}, \quad (5)$$
$$\text{where } j > 0, h = \text{floor}\left(j/2\right).$$

Algorithm 1: Pseudo-code for *Classic* and *HSB*

```
1:  Initialise each $Q_i$ with each bit $Q_{ij}=\pi/4$
2:  Initialise each $A_i$ with random strings
3:  **while** not termination condition **do**
4:      **for all** $i \in [1,p]$
5:          sample new $C_i$ from $Q_i$
6:          evaluate fitness of $C_i$ using a binary to real mapping
7:          **for each** $t \in [1,N]$
8:              **if** $f(A_i)$ is better than $f(C_i)$ **then** select a rotation
                    direction that would reinforce $A_{ij}$
9:              **else** select a rotation direction that would move away
                from $A_{ij}$
10:             **end if**
11:             update $Q_{ij}$ with rotation gate
12:         **end for**
13:         **if** $f(C_i)$ is better than $f(A_i)$ **then**
14:             $A_i = C_i$
15:         **end if**
16:     **end for**
17:     every $L$ iterations perform local migration
18:     every $G$ iterations perform global migration
19: **end while**
```

3. Real coded *QIEA* - *RCQIEA*

In order to apply *QIEA* to real-value problems, numerous attempts have been made to develop real *QIEA* (*rQIEA*) [40], so we chose to include *rQIEA* in this investigation. A simple attempt at this is shown in [34] where the rotation angles from the *Classic bQIEA* are re-interpreted as actual solutions. This approach was very simplistic and arbitrary, and completely failed in all of our initial testing. Searching for other *rQIEA* in the literature was made difficult by poorly described methods or suspect data. However, we found one algorithm called *RCQIEA*, presented in [41], to be well defined and decided to include it in our study, along with a modification.

Whereas *Classic* produces fresh solutions at each generation, *RCQIEA* stores and updates a candidate solution. *Classic* takes the inspiration of superposition and uses it to evolve a probability density function (pdf), as described by the probability angles for each bit. By not doing this directly, *RCQIEA* begins to move away from the original quantum metaphor. However, as we will describe shortly, the generation of new candidates through creep mutation, can be seen as using the candidate as a string of mean values for an evolving pdf.

At each iteration, a set of offspring O_j is generated from each individual's candidate C_i using creep mutation with variances stored in a string V_i. The values in V_i are stored as angles and transformed into a pair α_i and β_i in the same way as for the *Classic*. The offspring are generated in two subsets: one using α_i for the variances; and one using $\beta_i/5$, to allow for both fine and coarse searching. The offspring are tested for fitness and if one is found to be better than the current candidate, it replaces that candidate. Otherwise, a rotation gate is applied to the variance angles in the same way as in *Classic*, but with a rotation step given in Eq. (6).

$$\Delta\theta = \text{sgn}\left(\alpha\beta\right)\theta_0 \exp\left(\frac{|\beta|}{|\alpha| + \gamma}\right), \quad (6)$$

where α and β are the angles as defined in Eqs. (1) and (2), and θ_0 and γ are constants.

A cross-over operator is also applied during the evolution. For our investigation, we applied it four times during the course of each run ($G=N/4$), adopting the approach presented in [41]. The pseudo-code for the *rQIEA* presented here, is given in Algorithm 2.

3.1. Rotation magnitude analysis

In [41], the constants for (6) were specified as $\theta_0=0.4\pi$ and $\gamma=0.05$. In the testing, the *RCQIEA* performed well for many functions. However, we detected values of large magnitudes for $\Delta\theta$, which suggested a problem with the behaviour of the rotation gate. For example, if the angles are $\alpha=0.01$ and $\beta=0.99995$ (satisfying $\alpha^2 + \beta^2=1$), then (6) produces a value for $\Delta\theta$ in excess of 2.0e7. As a rotation angle in this context, such a magnitude for $\Delta\theta$ does not make sense, as it represents many complete rotations in one iteration. In effect this leads to somewhat random updates of the angle variables, and in turn, the variances for the creep mutation. A real example of these problematic delta values can be seen in Fig. 3.

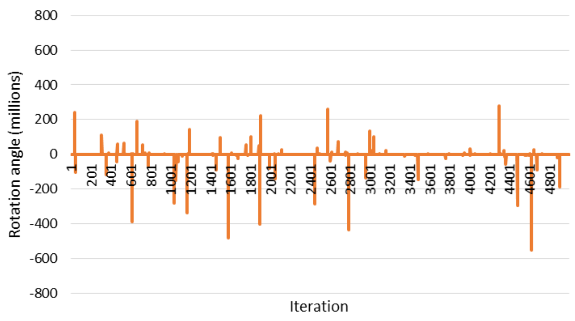


Fig. 3: An example of the $\Delta\theta$ values for the *RCQIEA* algorithm on the *Griewank* test function. The maximum magnitude should be π/2 but very large values can be observed.

Algorithm 2: Pseudo-code for *RCQIEA*

```
1.  Initialise the population size $p$, the maximum
    number of iterations $N$, and crossover
    frequency $G$
2.  Initialise each $C_i$, $V_i$ with random values
3.  Evaluate fitness $f(C_i)$ for each individual
4.  while not termination condition do
5.      for all $i \in [1, p]$
6.          construct two sets of offspring $O_j$ from
            $C_i$ using creep mutation from a normal
            distribution with variances $V_i$. One set
            uses the $\alpha_i$ angles and the other one the
            $\beta_i$ angles, both scaled for coarse and
            fine search respectively
7.          for each offspring $j$
8.              if $f(O_j)$ is better than $f(C_i)$ then
9.                  replace $C_i$ with $C_j$
10.             else apply rotation gate to $V_i$
11.             end if
12.         end for
13.     end for
14.     adjust coarse and fine search scale factors
        over course of run to move towards finer
        search at the end of the simulation
15.     every $G$ iterations perform crossover
        mutation
16. end while
```

3.2. Stepwise RQIEA - SRQEA

To alleviate this problem, we developed a modified version of the rotation gate, keeping the rest of the *RCQIEA* algorithm unchanged (see Algorithm 2). We call this modified algorithm *Stepwise Real QEA* (*SRQEA*). The change rotates the angles by a constant magnitude in the rotation gate, as shown in Eq. (7).

$$\Delta\theta = \operatorname{sgn}(\alpha\beta)\,\pi/250. \qquad (7)$$

This was motivated by making the update similar to the constant step size used in *Classic*. The value of *π/250* was chosen with respect to *Classic*'s step size of *π/100*. We made the step size smaller, because unlike *Classic* where the rotation angles control an absolute probability of getting zero or one, the angle variables for the *rQIEA* control a creep mutation variance. Since a larger spread of values is possible in this regime, we decided upon making the step size smaller. In early experimentation it yielded promising results, so we kept the value for the remainder of the investigation. It would be useful to compare different values for the step size in future work. Also, we kept *G=N/4* from [41] but other generation sizes could be investigated in the future.

4. Numerical experiments

Each algorithm was tested against several fitness functions. In accordance with the procedures outlined in [13], functions were tested with 10, 30, 50 and 100 dimensions (except for the real-world problems which had specific dimension requirements), and each optimization run was performed 51 times, unless otherwise stated. The termination criterion was set to a number of function evaluations of 10000 x number of dimensions, unless otherwise stated. Given that more than one function evaluation per generation was performed for the *rQIEA*, their generations per run were adjusted accordingly.

4.1. Test functions

Firstly, a set of traditional, basic functions, was taken from the first 13 functions presented in [8]. Additionally, a non-transformed basic version of *Schwefel 7* [25] was used when comparing to data published for three recent *QIEA* [37,42,43], and a basic two dimensional problem from [44], when comparing another *QIEA*. A second set of more complicated functions was added from the first 20 functions defined in the CEC-2013 specification [9]. These are based on the traditional functions but are highly modified and transformed, including application of rotations. It should be noted that both sets share one function in common – the *Sphere* function. We duplicate the presentation of the results for this function in order to be consistent when comparing to other published results. Finally, real-world problems from CEC-2011 [7] were added: *frequency modulated sound wave matching; atom configuration;* and *radar waveform parameter optimisation*.

The frequency modulated sound wave matching problem optimises the constants of Eq. (8), so that the output of the wave, measured for integer *t=[0,100]*, where *θ=2π/100*, matches the output of Eq. (9).

$$y(t) = a_1 \sin\left(\omega_1 t\theta + a_2 \sin\left(\omega_2 t\theta + a_3 \sin\left(\omega_3 t\theta\right)\right)\right) \quad (8)$$

$$y(t) = 1.0 \sin\left(5.0t\theta - 1.5 \sin\left(4.8t\theta + 2.0 \sin\left(4.9t\theta\right)\right)\right), \quad (9)$$

where *α* and *ω* are the constants to be optimised.

The *Lennard-Jones* atom potential configuration problem, aims to minimise the potential energy V_N of a set of *N* atoms with position $p_i = \{x_i, y_i, z_i\}$ according to Eq. (10).

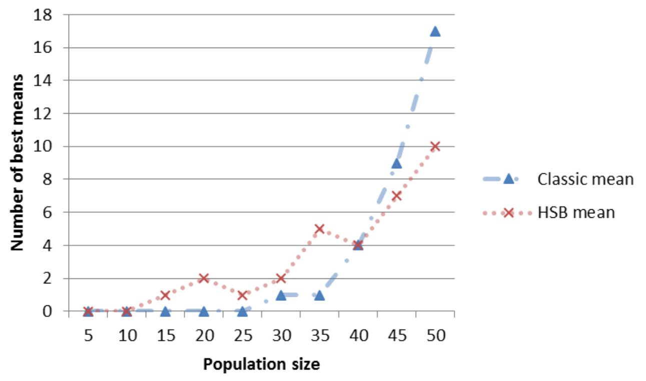
Finally, *the radar polyphase pulse design problem* seeks to minimise a function *f(x)* based upon set of *n* parameters $x=\{x_1,\ldots, x_n\}$ according to Eq. (11).

$$V_N(p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(r_{ij}^{-12} - 2r_{ij}^{-6}\right), \qquad (10)$$
$$r_{ij} = \left\| p_j - p_i \right\|_2 .$$

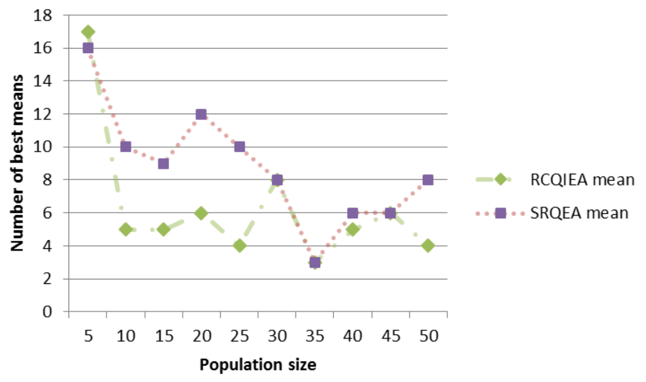
$$f(x) = \max\left\{\phi_1(x), \ldots, \phi_{2m}(x)\right\},$$
$$\phi_{2i-1}(x) = \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right), i = 1, \ldots, n,$$
$$\phi_{2i}(x) = 0.5 + \sum_{j=i+1}^{n} \cos\left(\sum_{k=|2i-j|+1}^{j} x_k\right), i = 1, \ldots, n-1, \qquad (11)$$
$$m = 2n - 1.$$

4.2. Population size analysis

Before conducting an extensive evaluation of the proposed methods, an investigation into choosing a suitable population size was conducted. An initial run for 30 dimensions was performed for the optimisation algorithms on the non-real world functions, with a series of different population sizes being used. The number of individuals ranged from 5 to 50, in increments of five, but the total number of functions evaluations was always kept to 300000. After running the simulations, the number of times an algorithm had a best performance (assessed just for that algorithm) was counted for each population size. A best performance occurred when it was the best, or equal best, minimum value or mean value for the fitness function of that optimisation algorithm.



a)



b)

Fig. 4: Population analysis for the *QIEA*: a) *Classic* and *HSB*; c) *RCQIEA* and *SRQEA*. For each algorithm, a simulation run was performed on the first 13 non-real world problems presented in [12] with 30 dimensions, with population sizes from 5 to 50. Then, for each algorithm in isolation, a count of best minimum and best mean values were produced for each population size (best as determined across all population sizes). The number of fitness evaluations was kept to 300000.

The results according to the best minimum and mean values found are shown in Fig. 4. Results for the *bQIEA Classic* and *HSB* are shown in Fig. 4a and results for the *rQIEA RCQIEA* and *SRQEA* are shown in Fig. 4b. Generally, the *bQIEA* performed better with higher population sizes, while the *rQIEA* were better with smaller population sizes. For *Classic* (Fig. 4a), the best minimum values were found more often with a population size of 50, with an additional peak at 20/25, while *HSB* (Fig. 4a) had a peak at 35/40 but reasonable performance from 25 to 50. When looking at the mean performance, both *bQIEA* improved with increasing population size, with the best being 50 for both. After combining these results, we chose to proceed with 50 individuals for both *bQIEA* algorithms in the later simulations and analysis. These results suggest bQIEA are biased towards exploitation and therefore require a larger population size to achieve good exploration.

For both *rQIEA*, the results (Fig. 4b) were very clear – a population size of five performed the best for both minimum and mean values. *RCQIEA* had a sharp drop-off in performance above five, while *SRQEA* had a smoother decline with increasing population size. Based upon these results, a population size of five was chosen for both *rQIEA*. In contrast to the results for the *bQIEA*, these results suggest the *rQIEA* have relatively good exploration, so benefit from a small population in order to improve exploitation by increasing the number of function evaluations per individual.

4.3. Step size analysis

For all the *QIEA* tested here, step size $\Delta\theta$ is a parameter to be specified. As described in sections 2 and 3, initial values for *HSB* and *SRQEA* respectively were chosen to be $\pi/100$ and $\pi/250$, with respect to the original $\pi/100$ in [29]. However, we did conduct a preliminary investigation into the effects of varying these parameters, with the results presented in Table 1. The best parameter values for *SRQEA* varied greatly across the different fitness functions, and somewhat symmetrically around $\pi/250$. We therefore continued to use this default value for the remainder of our investigations. The results for *HSB* did lend some support to slightly larger step sizes than $\pi/100$, but as the results were also mixed, we decided to be consistent with the original value from [29] for subsequent tests.

Table 1: Best step size values (given as a coefficient of π) for *HSB* and *SRQEA* for the CEC-2013 functions on 30 dimensions. Best values (tested across a range) are selected according to which produced the minimum fitness over 30 runs. When testing SRQEA, four test functions (labelled SP) obtained threshold zero minimum values for all step sizes tested. Therefore, to pick a preferred value, we used the one that produced the best SP value (described in section 4.4).

Function – 30 dimensions	HSB / π	SRQEA / π
Sphere	0.0164	0.001 (SP)
Rotated high conditioned elliptic	0.0092	0.0064
Rotated bent cigar	0.0110	0.0010
Rotated discuss	0.0038	0.0019
Different powers	0.0155	0.0013 (SP)
Rotated Rosenbrock	0.0191	0.0010
Rotated Schaffers F7	0.0083	0.0064
Rotated Ackley	0.0191	0.0010
Rotated Weierstrass	0.0128	0.0052
Rotated Griewank	0.0137	0.0070
Rastrigin	0.0155	0.001 (SP)
Rotated Rastrigin	0.0137	0.0010
Non continuous rotated Rastrigin	0.0056	0.0010
Schwefel 7	0.0146	0.0040
Rotated Schwefel 7	0.0173	0.0019
Rotated Katsuura	0.0101	0.0046
Lunacek bi-Rastrigin	0.0182	0.0031 (SP)
Rotated Lunacek bi-Rastrigin	0.0164	0.0052
Rotated expanded Griewank Rosenbrock	0.0155	0.0061
Rotated expanded Schaffers F6	0.0146	0.0010

4.4. Performance metrics

4.4.1. Summary statistics
To present a basic analysis and compare across publications, summary information is generated from error values (from the known minimum value) or absolute values if the global minimum is unknown. From the raw data, simple statistical measures such as minimum, mean and standard deviations are calculated and summarised, with lower values for each being preferred in the comparisons.

4.4.2. Success Rates
Using metrics introduced in [45], a success rate and measure of time taken by the run to succeed (converging to a minimum) are calculated. Success Rate (SR) is calculated as the number of successful runs divided by the total number of runs. A run is regarded as successful if it finds an error below a predefined threshold.

4.4.3. Success Performance
To measure the speed at which an algorithm obtains good results, a metric called Success Performance (SP) is calculated. It is defined in Eq. (12).

$$SP = \frac{SNFEs}{SR} \qquad (12)$$

where *SNFE* is the average number of function evaluations required by each successful run to reach the tolerance. A lower value of SP is preferred because it indicates a better combination of speed and consistency for the algorithm.

4.4.4. Timeline plots
In order to analyse the behaviour of the algorithms, graphical representations of their evolution are produced for every test function. Across all runs, for each iteration the mean error is calculated and plotted, so that the behaviour, with respect to the number of function evaluations, can be compared directly between the algorithms, and the time is normalised in the [0, 1] range.

4.4.5. Population diversity
To analyse the behaviours deeper, we present plots on showing the evolution of population diversity. We use a standard metric based on mean individual variance from around the population mean position (position in terms of *Qbit* string representation in Euclidean space) [46]. It is given in Eq. (13).

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$
$$D = \frac{1}{mn}\sum_{j=1}^{n}\sum_{i=1}^{m}\left[x_{ij} - \bar{x}_j\right]^2 \tag{13}$$

where *i* is the individual, *j* is the dimension, *m* is the number of individuals, *n* is the number of dimensions, *x* is the *Qbit* string, $\bar{x}_j$ is the mean position and *D* is the diversity.

4.4.6. Empirical cumulative probability distribution
Performance across all functions is summarised using the empirical cumulative probability distribution function (ECDF) method presented in [47]. An ECDF is constructed by firstly determining the performance of each algorithm on each test function, by comparing its mean error *ME* with the mean error achieved by the best algorithm, and formulating a normalized mean error *NME* (Eq. (14)). Then, the distribution is formed by counting, for each value x in the domain of the distribution, how many normalized means (across all test functions) were obtained below x (Eq. (15)). Normalizing and plotting these values produces a graph where superior algorithms reach the top of the chart faster than less well performing algorithms. In this analysis, all the test functions were included, as well as additional graphs for subsets (traditional, CEC-2013 and real-world).

$$NME_{A,f} = \frac{ME_{A,f}}{ME_{best,f} + 1}, \tag{14}$$

$$ECDF(x) = \frac{1}{n_A \times n_f}\sum_{i=1}^{n_A}\sum_{j=1}^{n_f} I\left(NME_{i,j} \le x\right), \tag{15}$$

where *A* and *f* are the optimisation algorithm and the test function index respectively, n_A and n_f are the number of algorithms and test functions respectively, and *I* is 1 if the condition is true, otherwise 0.

4.4.7. Computation time
Although results are compared with the same number of function evaluations, the computational overhead of each algorithm must be taken into account. Accurate time measurements are presented, showing the average function evaluation overhead, and generation overhead.

5. Results and Discussion

Examples of methods used to optimise CEC-2013 problems include *Particle Swarm Optimization* [15], *Adaptive Differential Evolution* [19,48,49], *Mean Variance Mapping* [50] and *GA* [17]. The methods for optimisation of the traditional test functions, covered in this work, include *Evolutionary Programming* [12], *Particle Swarm Optimization* [51], *GA* [52], and *Hybrid Bee Colony/QEA* [53]. This section presents the *bQIEA* and *rQIEA* results that we produced.

5.1. Functionality of the tested QIEA

Firstly, we examined the suitability of the four *QIEA*, tested to be used as optimisation algorithms for real-value problems. In order to be useful, they must find solutions close to the optimum, as seen by reaching small error values. We start by looking at the performance on the traditional test functions which were tested at 10, 30, 50 and 100 dimensions. Minimum, mean and standard deviation data are presented as functions 0-13 in Table 2 and Table 3 for 50 and 100 dimensions respectively.

Table 2: Summary statistics for the 13 traditional and 19 CEC-2013 test functions (duplicated 14 *Sphere* removed) with 50 dimensions and 500000 function evaluations. For each of the four optimization algorithms, the minimum, mean and standard deviation of the error values are presented after 51 runs. Best values are highlighted in bold type.

Traditional and CEC-2013 functions 50 Dimensions	***bQIEA***						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
01 Sphere	1.35E+03	3.35E+03	1.07E+03	1.81E+02	1.63E+03	7.16E+02	3.01E-04	5.81E-04	1.79E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
02 Schwefel-2.22	9.12E+01	1.67E+02	3.90E+01	4.92E+01	1.23E+02	3.17E+01	7.98E-02	1.08E-01	1.30E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
03 Schwefel-1.2	5.44E+05	2.20E+06	1.01E+06	1.48E+05	7.98E+05	4.56E+05	2.03E-01	4.26E-01	2.13E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**
04 Schwefel-2.21	2.16E+01	3.63E+01	6.13E+00	1.83E+01	2.44E+01	4.01E+00	1.81E-01	3.05E-01	4.89E-02	**2.00E-02**	**3.29E-02**	**7.54E-03**
05 Rosenbrock	9.29E+06	2.81E+08	1.78E+08	7.79E+06	9.17E+07	7.77E+07	9.37E+00	1.27E+02	5.77E+01	**4.49E-02**	**4.34E+01**	**3.09E+01**
06 Step	1.22E+03	3.29E+03	1.41E+03	6.51E+02	1.72E+03	6.53E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
07 Quartic	3.76E+06	6.42E+07	6.02E+07	6.57E+05	1.51E+07	1.33E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
08 Schwefel-2.26	1.22E+02	3.21E+02	1.30E+02	4.15E+01	1.88E+02	8.73E+01	3.17E-05	6.60E-05	2.20E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
09 Basic Rastrigin	7.14E+01	1.04E+02	1.44E+01	3.06E+01	6.18E+01	1.06E+01	1.56E-04	2.89E-04	8.13E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
10 Basic Ackley	1.59E+01	1.93E+01	6.07E-01	1.42E+01	1.72E+01	1.12E+00	1.02E-02	1.66E-02	3.10E-03	**0.00E+00**	**5.63E-07**	**3.26E-06**
11 Basic Griewank	1.07E+01	2.92E+01	1.22E+01	3.64E+00	1.53E+01	5.86E+00	4.01E-04	**8.45E-03**	**9.66E-03**	**0.00E+00**	1.48E-02	2.59E-02
12 Penalised-1	1.04E+06	4.77E+07	3.66E+07	8.72E+01	1.09E+07	1.74E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
13 Penalised-2	1.43E+07	1.38E+08	9.05E+07	5.21E+05	2.95E+07	3.73E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	5.30E+07	1.17E+08	4.11E+07	4.55E+07	1.02E+08	3.28E+07	5.84E+06	1.15E+07	3.01E+06	**1.55E+06**	**2.96E+06**	**6.43E+05**
16 Rotated bent cigar	1.38E+10	3.21E+10	8.83E+09	4.85E+09	1.90E+10	1.04E+10	1.18E+03	6.45E+06	2.26E+07	**4.98E-02**	**1.58E+05**	**1.08E+06**
17 Rotated discus	5.44E+04	8.77E+04	1.68E+04	**1.02E+04**	**2.40E+04**	**7.45E+03**	1.38E+05	1.92E+05	2.88E+04	1.14E+05	1.75E+05	2.47E+04
18 Different powers	5.98E+02	4.90E+03	3.35E+03	9.52E+01	1.14E+03	1.46E+03	1.57E-03	4.14E-03	1.94E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	1.12E+02	3.10E+02	1.01E+02	1.04E+02	2.72E+02	9.86E+01	2.98E+01	4.51E+01	**3.43E+00**	**2.38E+01**	**4.19E+01**	7.54E+00
20 Rotated Schaffers-F7	1.44E+02	1.83E+02	**2.18E+01**	**1.21E+02**	**1.78E+02**	2.47E+01	1.79E+02	2.54E+02	1.22E+02	1.47E+02	2.46E+02	9.28E+01
21 Rotated Ackley	2.11E+01	2.12E+01	4.08E-02	2.10E+01	2.12E+01	**3.58E-02**	2.10E+01	2.11E+01	3.74E-02	**2.10E+01**	**2.11E+01**	4.68E-02
22 Rotated Weierstrass	4.46E+01	5.27E+01	4.77E+00	**4.07E+01**	**5.27E+01**	4.47E+00	5.71E+01	6.34E+01	**3.36E+00**	6.16E+01	6.74E+01	3.76E+00
23 Rotated Griewank	5.36E+02	1.05E+03	2.68E+02	4.03E+02	8.27E+02	2.22E+02	1.54E+00	2.25E+00	3.02E-01	**2.71E-02**	**1.31E-01**	**4.96E-02**
24 Rastrigin	7.36E+01	1.33E+02	2.56E+01	5.48E+01	7.99E+01	1.82E+01	5.83E-04	1.10E-03	3.38E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
25 Rotated Rastrigin	2.41E+02	**3.69E+02**	**7.14E+01**	**2.33E+02**	3.86E+02	8.38E+01	3.58E+02	6.08E+02	1.25E+02	4.60E+02	6.85E+02	1.38E+02
26 NC rotated Rastrigin	3.95E+02	**5.23E+02**	**7.68E+01**	**3.51E+02**	5.60E+02	1.15E+02	4.71E+02	6.30E+02	9.74E+01	5.01E+02	6.89E+02	1.01E+02
27 Schwefel-7	3.85E+02	1.17E+03	3.62E+02	1.43E+02	5.36E+02	2.46E+02	**2.96E-02**	**8.57E-02**	**2.41E-02**	9.99E-02	6.71E-01	2.94E-01
28 Rotated Schwefel-7	5.68E+03	7.73E+03	9.96E+02	5.12E+03	7.33E+03	9.92E+02	**4.37E+03**	6.25E+03	7.15E+02	4.69E+03	**6.22E+03**	**6.23E+02**
29 Rotated Katsuura	**8.49E-01**	2.02E+00	5.86E-01	1.26E+00	1.91E+00	4.11E-01	8.74E-01	**1.64E+00**	**3.49E-01**	8.93E-01	1.83E+00	4.41E-01
30 Lunacek bi-Rastrigin	1.31E+02	2.68E+02	6.63E+01	9.44E+01	1.65E+02	4.14E+01	3.82E-02	9.98E-02	3.51E-02	**0.00E+00**	**1.96E-04**	**1.40E-03**
31 R Lunacek bi-Rastrigin	3.41E+02	6.49E+02	1.27E+02	4.52E+02	6.51E+02	1.08E+02	**3.05E+02**	**4.80E+02**	**7.87E+01**	4.53E+02	6.12E+02	9.30E+01
32 RE Griewank Rosen.	9.91E+01	5.90E+02	7.94E+02	7.21E+01	4.55E+02	5.85E+02	**5.73E+01**	**1.45E+02**	**4.66E+01**	1.46E+02	2.91E+02	6.26E+01
33 RE Schaffers-F6	1.51E+01	1.93E+01	2.47E+00	**1.51E+01**	**1.84E+01**	1.95E+00	2.05E+01	2.43E+01	**5.97E-01**	1.90E+01	2.44E+01	7.68E-01

Table 3: Summary statistics for the 13 traditional and 19 CEC-2013 test functions (duplicated 14 Sphere removed) with 50 dimensions and 500000 function evaluations. For each of the four optimization algorithms, the minimum, mean and standard deviation of the error values are presented after 51 runs. Best values are highlighted in bold type.

Traditional and CEC-2013 functions 100 Dimensions	***bQIEA***						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
01 Sphere	4.14E+03	7.66E+03	2.10E+03	2.74E+03	4.86E+03	1.31E+03	7.23E-04	1.12E-03	2.16E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
02 Schwefel-2.22	2.92E+02	4.18E+02	6.37E+01	2.14E+02	3.08E+02	5.11E+01	1.66E-01	2.14E-01	1.99E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
03 Schwefel-1.2	1.26E+07	2.52E+07	7.84E+06	4.69E+06	1.07E+07	3.88E+06	1.65E+00	3.29E+00	1.01E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
04 Schwefel-2.21	3.41E+01	4.37E+01	4.97E+00	2.37E+01	3.12E+01	3.76E+00	5.96E-01	8.21E-01	1.04E-01	**1.82E-01**	**2.70E-01**	**4.67E-02**
05 Rosenbrock	1.43E+08	6.40E+08	2.70E+08	8.49E+07	2.87E+08	1.36E+08	1.49E+01	2.09E+02	7.68E+01	**2.77E-02**	**3.49E+01**	**2.75E+01**
06 Step	3.54E+03	8.47E+03	1.97E+03	2.02E+03	4.10E+03	1.08E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
07 Quartic	1.29E+08	3.33E+08	1.43E+08	1.88E+07	1.19E+08	6.35E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
08 Schwefel-2.26	4.78E+02	8.93E+02	2.11E+02	3.11E+02	5.52E+02	1.55E+02	8.42E-05	1.46E-04	3.26E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
09 Basic Rastrigin	1.86E+02	2.28E+02	2.36E+01	1.23E+02	1.58E+02	2.00E+01	3.54E-04	5.79E-04	1.57E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
10 Basic Ackley	2.00E+01	2.00E+01	1.51E-02	1.62E+01	1.83E+01	8.65E-01	1.33E-02	1.68E-02	2.09E-03	**0.00E+00**	**1.19E-06**	**8.12E-06**
11 Basic Griewank	3.66E+01	7.20E+01	1.67E+01	2.22E+01	4.31E+01	1.17E+01	4.40E-04	**8.56E-03**	**1.16E-02**	**0.00E+00**	2.16E-02	4.74E-02
12 Penalised-1	1.81E+07	1.68E+08	9.57E+07	2.47E+06	4.10E+07	4.19E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
13 Penalised-2	8.90E+07	3.05E+08	1.65E+08	1.36E+07	1.03E+08	6.74E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	1.25E+08	3.01E+08	8.64E+07	1.46E+08	2.49E+08	5.79E+07	1.47E+07	2.16E+07	3.80E+06	**4.41E+06**	**6.54E+06**	**9.97E+05**
16 Rotated bent cigar	7.46E+10	2.32E+11	1.62E+11	4.84E+10	1.22E+11	5.84E+10	1.15E+03	8.97E+03	3.31E+04	**2.17E-03**	**4.02E+02**	**8.35E+02**
17 Rotated discus	1.41E+05	1.95E+05	2.65E+04	**2.87E+04**	**5.31E+04**	**1.10E+04**	2.51E+05	3.64E+05	4.48E+04	2.29E+05	3.21E+05	4.25E+04
18 Different powers	2.82E+03	1.19E+04	8.19E+03	3.91E+02	3.14E+03	2.87E+03	2.49E-03	5.88E-03	2.03E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	6.85E+02	1.07E+03	2.64E+02	5.83E+02	9.54E+02	1.92E+02	9.29E+01	1.94E+02	**3.46E+01**	**2.50E+01**	**1.45E+02**	3.63E+01
20 Rotated Schaffers-F7	2.26E+02	9.23E+02	7.95E+02	**2.14E+02**	**5.38E+02**	**3.61E+02**	3.46E+02	2.01E+04	2.00E+04	4.51E+02	2.06E+05	3.78E+05
21 Rotated Ackley	2.13E+01	2.14E+01	**2.33E-02**	2.13E+01	2.13E+01	2.44E-02	2.12E+01	2.13E+01	2.48E-02	**2.12E+01**	**2.13E+01**	3.94E-02
22 Rotated Weierstrass	**1.04E+02**	1.23E+02	9.04E+00	1.05E+02	**1.23E+02**	7.27E+00	1.39E+02	1.49E+02	**5.16E+00**	1.41E+02	1.51E+02	5.39E+00
23 Rotated Griewank	1.18E+03	2.46E+03	6.31E+02	1.02E+03	1.97E+03	4.14E+02	2.48E+00	3.53E+00	3.60E-01	**9.87E-03**	**7.15E-02**	**3.82E-02**
24 Rastrigin	2.34E+02	3.19E+02	4.12E+01	1.48E+02	2.11E+02	2.97E+01	1.27E-03	2.07E-03	5.22E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
25 Rotated Rastrigin	7.33E+02	**9.81E+02**	**1.48E+02**	**7.21E+02**	1.04E+03	1.75E+02	1.15E+03	1.78E+03	3.61E+02	1.30E+03	1.88E+03	2.91E+02
26 NC rotated Rastrigin	1.06E+03	**1.35E+03**	**1.64E+02**	**9.65E+02**	1.38E+03	1.77E+02	1.33E+03	1.78E+03	2.19E+02	1.57E+03	2.01E+03	2.40E+02
27 Schwefel-7	1.67E+03	2.73E+03	5.43E+02	8.09E+02	1.37E+03	4.08E+02	**8.69E-02**	**1.63E-01**	**4.62E-02**	5.10E-01	1.26E+00	4.23E-01
28 Rotated Schwefel-7	1.40E+04	1.70E+04	1.32E+03	1.27E+04	1.65E+04	1.60E+03	**1.09E+04**	**1.35E+04**	1.02E+03	1.12E+04	1.38E+04	**8.66E+02**
29 Rotated Katsuura	**1.44E+00**	2.63E+00	5.62E-01	1.47E+00	2.50E+00	4.81E-01	1.71E+00	**2.31E+00**	**3.07E-01**	1.86E+00	2.56E+00	3.63E-01
30 Lunacek bi-Rastrigin	4.40E+02	6.00E+02	9.39E+01	2.74E+02	4.18E+02	7.67E+01	1.21E-01	1.96E-01	4.34E-02	**0.00E+00**	**4.77E-04**	**2.41E-03**
31 R Lunacek bi-Rastrigin	1.28E+03	1.82E+03	2.66E+02	1.30E+03	1.80E+03	2.69E+02	**1.22E+03**	**1.75E+03**	**2.37E+02**	1.71E+03	2.43E+03	3.42E+02
32 RE Griewank Rosen.	1.05E+03	7.79E+03	6.89E+03	1.09E+03	5.83E+03	4.99E+03	**3.32E+02**	**5.28E+02**	**1.04E+02**	7.30E+02	1.10E+03	2.02E+02
33 RE Schaffers-F6	**4.95E+01**	**4.98E+01**	2.50E-01	**4.95E+01**	4.98E+01	2.46E-01	4.95E+01	5.00E+01	**9.90E-02**	**4.95E+01**	4.99E+01	2.22E-01

These functions are reasonably smooth, at least locally, and therefore obtaining a good error score will require good exploitation abilities of the algorithm. In section 2 we highlighted the difficulties for *Classic* in optimising the LSBs, and we would expect this to be reflected in poor minimum values as the exploitation would be hampered. Most solutions had errors of magnitude above 1e-01, the only exceptions being the 10 dimensional *Schwefel-2.26*, *Griewank* and *Penalised-1* functions. Interpreting raw error values is difficult because it relates to the numerical properties of the fitness function. For example, the *Rosenbrock* has a constant multiplication factor of 100, and for the 10-dimension case achieved 1.15e02 which is perhaps not as bad as it would first appear. However, as the number of dimensions increases, the performance becomes obviously poorer, with four minima of magnitude over 1e06 at 50 dimensions and five at 100 dimensions. Means performance for all of the dimensions tested follows the same pattern in general, although there are some large discrepancies for the 10-dimension batch. These are a mean of 2.32e06 compared to a minimum of 1.15e02 for *Rosenbrock*, 2.27e05 versus 4.26e01 for *Quartic*, and 4.62e05 and 1.11e06 versus 0e00 for *Penalised-1* and *2* respectively. This implies, at least for low dimensions, that running *Classic* several times is a necessity.

For every test function in the traditional batch, *HSB* had equal or better minimum values than *Classic*, apart from the 10-dimension *Rosenbrock* function. Magnitudes were generally similar, except for *HSB* achieving a minimum of 8.72e01 for 50 dimension *Penalised-1* compared to 1.03e06 for *Classic*. *HSB* had superior mean performance for all of traditional test functions. The consistently better performance suggests both that the LSB problems of *Classic* hampered its performance, and that our tested solution of limiting the LSB probability saturation was successful.

Despite apparent functional performance by the *bQIEA*, the two *rQIEA* were substantially better. The worst performance for the *rQIEA* was for *RCQIEA* on the 30-dimension *Rosenbrock* with a minimum of 1.65e01, but most minima had magnitudes of less than 1e-01. *RCQIEA* found solutions smaller than 1e-08 (clamped to 0.00 in the results) for *Step*, *Quartic*, *Penalised-1* and *Penalised-2* in all tested dimensions. Despite *RCQIEA* performing well on these test functions, it was eclipsed by *SRQEA*. With the exception of *Schwefel*-2.21 and *Rosenbrock*, it obtained clamped 0.00 results for all of the functions, in all dimensions. Even for *Schwefel*-2.21 and *Rosenbrock* it had the best performance across the *QIEA* tested. The superior performance of the real algorithms over their binary counterparts is unsurprising, given the application to real-value problems, but the superior performance of *SRQEA* justifies our modification of the rotation gate function for these functions.

As CEC-2013 are a set of real-value problems, some being modified versions of the functions from the traditional set tested here, we predicted that a similar pattern of results would be generated, with the *rQIEA* dominating the *bQIEA*. Although *HSB* outperformed

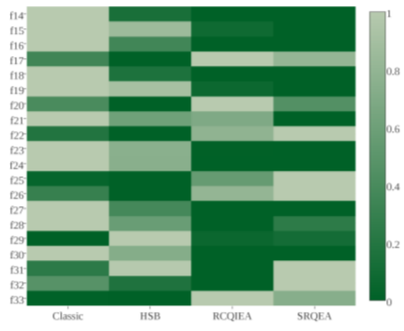
Classic, and *SRQEA* outperformed *RCQIEA*, the performance of the *bQIEA* compared to the *rQIEA* was very different from its previous performance (see Table 2 and Table 2 functions 15-33 for 50 and 100 dimensions respectively).

For several of the test functions - Rotated Discus, Rotated Schaffers-F7, Rotated Weierstrass, Rotated Rastrigin, Non-continuous Rotated Rastrigin, Rotated Schwefel 7, Rotated Katsuura, Rotated Expanded Grienwank-Rosenbrock and Rotated Expanded Schaffers-F6, one of the *bQIEA* had the best performance for one or more dimensions tested. When the *bQIEA* performed best, the *rQIEA* approached a similar order of magnitude, but when one of the *rQIEA* gave the best result, they often performed considerably better than the *bQIEA* (for example, on the 100 dimension Rotated Bent Cigar, *SRQEA* achieved 2.17e-03 compared to 7.46e10 and 4.84e10 for *Classic* and *HSB* respectively). Nevertheless, the positive results of the *bQIEA* are significant and surprising, given that they can outperform the *rQIEA* on some real-value benchmark functions.

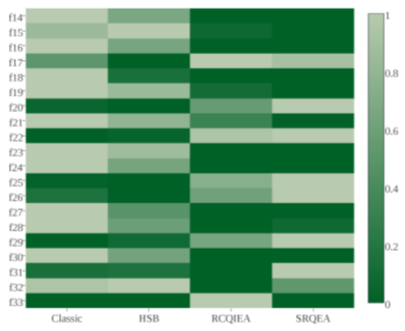
The CEC-2013 functions are highly manipulated versions of traditional basic functions (many based on the traditional test functions used in this paper). The manipulations include rotations, scalings and non-linear transforms. We hypothesise that it is these transformations that allow the *bQIEA* to perform well and suggest that this could happen in one of two possible ways. Firstly, the transformations may increase the nonlinear interactions between dimensions, producing a fitness landscape that is very rough, and therefore more resembling a discrete space at scales above the very small. These search spaces may be suited to the binary methods presented here, possibly possessing similarities to the combinatorial problems that *bQIEA* have been successful with (e.g., *Knapsack* [29]). Alternatively, the search pattern may be the key. In the *rQIEA*, the search space is traversed using creep mutations with distances drawn from a normal distribution, while the movement in the *bQIEA* is performed using multi-scaled jumps as the bits flip between zero and one and move the search to an adjacent binary partition at the scale of the significance of the bit. This binary space partitioning could reflect, to some degree, the underlying structure of the search spaces.

For the *CEC-2013* set of test functions, the *bQIEA* achieved several minimum scores with a magnitude of 1e02 or less and, given that the test functions often contain large constants (1e06), it could be argued that they performed better on the more difficult test functions than on the traditional set of functions. It would be interesting to see if this scales, so that the *bQIEA* have increasingly better relative performance as the fitness landscape becomes more complex. *HSB* appears to scale better than *Classic*, achieving five best performances across all four QIEA for 50 dimensions, compared to one for *Classic* although for 100 dimensions the balance was closer – five and three respectively.

Although *SRQEA* performed the greatest, in terms of number of best minimum values found and the ability to find threshold zero error values for some functions (which none of the other algorithms managed to do), when looking at the general performance across all of the functions and algorithms, the picture was somewhat more mixed. A heat map of best minimum values, scaled relatively from the best performing algorithm to the worst on each test function, is presented in Fig. 5. For 50 dimensions (Fig. 5a), judging by the number of darker rectangles, *RCQIEA* performs well, arguably outperforming *SRQEA*. From the raw data in Table 2, it can be seen that when the performances of the *rQIEA* are close, *SRQEA* produces better results than *RCQIEA*, but this is not generally noticeable in the heat map, where the larger degrees of magnitude produced by the *bQIEA* obscure the *rQIEA* differences. Although patterns persist for 100 dimensions (Fig. 5b), *Classic* demonstrates a closer performance to *HSB* when the *bQIEA* outperform the *rQIEA*. Summarising the raw data and the heat map, it can be said that *RCQIEA* had a slightly better average performance for 50 dimensions but *SRQEA* was able to produce much better individual scores for some functions, and was better overall for 100 dimensions. The more random nature of the rotation gate of *RCQIEA* may produce desirable search characteristics for the *CEC-2013* test functions, at the expense of more exploitation.



a)



b)

Fig. 5: Heat map of best convergence to a minimum by the *QIEA* on the CEC-2013 test functions on a) 50 dimensions and b) 100 dimensions. For each test function, the relative performance for each algorithm is plotted, with a green (zero) rectangle indicating best performance, and a light-green (one) rectangle indicating worst performance.

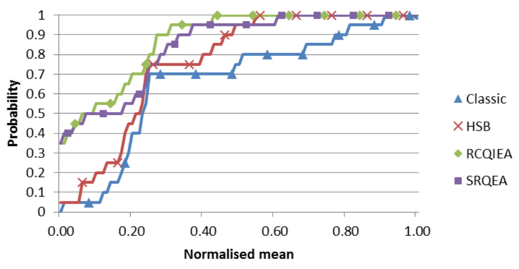
For the *CEC-2011* real-world problems, converging to the minima was best for the *rQIEA* (Table 4), with *SRQEA* having the best scores for three of the functions. However, for the *Radar Polly Phase* problem *HSB* had the best result, and shared the number of best means equally with *SRQEA*. The nested functions of the *Frequency Modulation* and *Radar Polly Phase* problems suggest a highly nonlinear search space, so these results are consistent with our findings and interpretations of the performance of the *bQIEA* on the *CEC-2013* functions.

Table 4: Summary statistics for *CEC-2011* real world problems. For each of the four optimization algorithms, the minimum, mean and standard deviation of the error values are presented after 51 runs. Best values are highlighted in bold type. Function evaluations were limited to 150000.

Function	*bQIEA*						*rQIEA*					
	Classic			*HSB*			*RCQIEA*			*SRQEA*		
	Min	Mean	Std dev	Min	Mean	Std dev	Min	Mean	Std dev	Min	Mean	Std dev
FM	4.42E-02	1.34E+01	6.11E+00	3.01E-03	**1.07E+01**	6.99E+00	2.71E-04	1.57E+01	5.78E+00	**0.00E+00**	1.70E+01	**4.68E+00**
L-J5	-1.21E+01	-9.62E+00	1.49E+00	-1.22E+01	-1.03E+01	1.55E+00	-1.27E+01	-1.18E+01	1.03E+00	**-1.27E+01**	**-1.21E+01**	**1.02E+00**
L-J10	-2.72E+01	-1.80E+01	4.03E+00	-2.67E+01	-1.95E+01	**3.79E+00**	-3.08E+01	-2.26E+01	3.87E+00	**-3.18E+01**	**-2.41E+01**	4.23E+00
Radar	1.58E+00	2.00E+00	**1.97E-01**	**1.40E+00**	**1.91E+00**	2.02E-01	1.50E+00	2.00E+00	2.31E-01	1.59E+00	2.11E+00	2.10E-01

Finally, we present a summary of algorithms' mean performance across multiple test functions in Fig. 6. The plots show a cumulative normalised count of how many functions possess a normalised mean performance for that algorithm, below a given value. The sooner the plot reaches 1.0 in the vertical axis, the better the algorithm performs (as this indicates a high probability of achieving low mean error values).

a)



b)

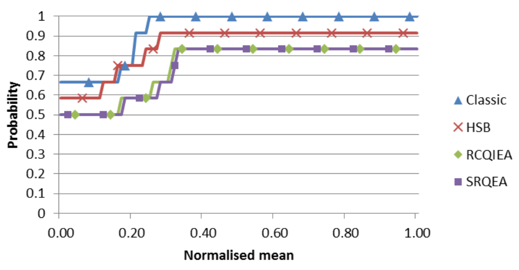


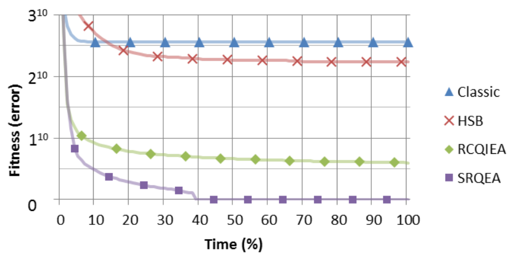
Fig. 6: Empirical cumulative probability distribution function of mean errors across a) CEC-2013 100 dimensions, and b) real-world all test functions, comparing the four *QIEA*. The horizontal axis shows normalised mean score, and the vertical axis shows cumulative probability. The faster the approach to 1 in the vertical direction, the better the performance.

The best performance on the traditional test functions (not shown) is dominated by the two *rQIEA* methods, which can also be seen for all of the test functions taken together (not shown), with *Classic* performing poorly for both of those cases. However, for the *CEC-2013* functions 100 dimensions *HSB* is much closer (Fig. 6a), catching up sooner with the *rQIEA* in the plot, although it starts with poorer results, indicating a low probability of producing very low mean scores across the function set. The performance of *RCQIEA* compared to *SRQEA* for CEC-2013 is in line with the results presented in the heat maps (Fig. 5). *SRQEA* underperforms *RCQIEA* for a significant range, suggesting the improved ability to find minimum values negatively impacts mean scores. For the real-world test functions (Fig. 6b), the situation is completely reversed, with *Classic* performing the best, followed by *HSB*.

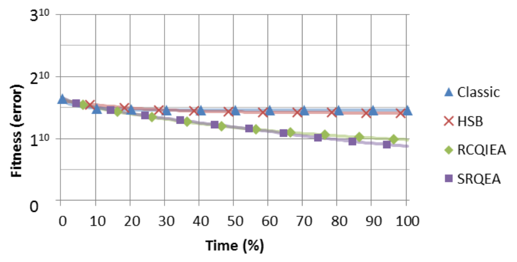
Summarising the ECDF and the results given in the tables, we can conclude that, although the *rQIEA* have superior best performance (minimum values found), the *bQIEA* algorithms do have good mean performance, often superior to their real-value counterparts. Again, it is with the more complicated *CEC-2013* and real-world *CEC-2011* functions that the *bQIEA* perform at their best, often outperforming the *rQIEA*.

5.2. Evolution properties of the QIEA

Mean error values per generation (averaged across the 51 runs) are shown for two functions in Fig. 7. For most functions, *Classic* outperformed *HSB* early on the evolution, but tends to stall earlier and is generally overtaken by *HSB* at around the 10 - 30% (of the total number of generations) time point (for example, see the *Different Powers* function timeline in a). This gives additional support to our argument that *Classic* was prematurely converging when applied to real-value problems, and justifies our approach when formulating the *HSB* adaptation. However, it should also be noted that *HSB* also usually approaches an approximately zero gradient relatively early on (50% of time or less), implying there is further need to improve premature convergence.



a)



b)

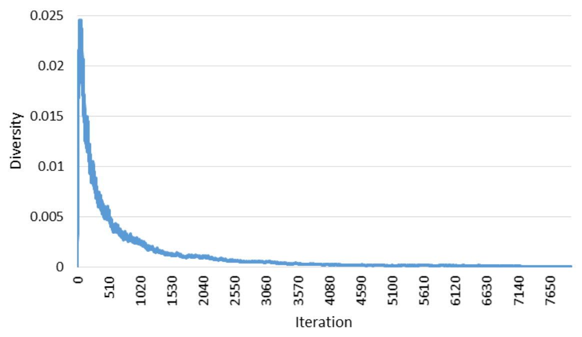
Fig. 7: Timeline evolution of mean error values. The mean error for each generation was calculated across each of the 51 runs, for every test function, and plotted for 100 dimensions on a) *Different Powers,* and b) *Schwefel-2.21*. Each graph plots time normalized evolutions, comparing the relative performance of the optimization algorithms.

For the majority of cases where *SRQEA* outperformed *RCQIEA*, their early performances were very similar, but *SRQEA* would establish a lead from typically the 5-30% time mark (see *Different Powers* in Fig. 7a). We interpret this as indicating that our corrected rotation formula allowed a more refined search in later stages. Both *rQIEA* demonstrated a clear non-zero gradient at the end of the timeline in several of the plots (such as Fig. 7b). This suggests they are capable of finding significantly better results if the algorithm is run for longer. As the plots display the fitness to the 10th root, this is relevant for fine convergence to the optimal value, indicating room for improvement of precision.

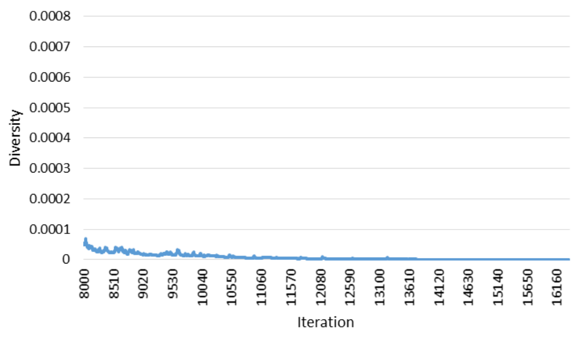
In order to compare the speed of evolution for the *QIEA* on functions for which zero minima were obtained, success rates (SR) and success performances (SP) were calculated for *RCQIEA* and *SRQEA* for those test functions, using four threshold values: 1e-02, 1e-04, 1e-06 and 1e-08. The data are presented in Table 5. In almost all cases, *SRQEA* outperformed *RCQIEA*, with the only exception being the success rate for the basic *Griewank* function at the 1e-02 threshold. The SP metric gives the mean number of function evaluations per success, adjusted in order to penalise low success rates. In conclusion, the data show that *SRQEA* provides superior success rates, and quicker convergence than *RCQIEA*.

Table 5: Success rates (SR) and success performance (SP) for the test functions at 30 dimensions, which reached a threshold of 1e-8 by one of the quantum algorithms, for different success thresholds: 1e-2; 1e-4; 1e-6; and 1e-8. SR ranges from 0 (no successes) to 1 (all runs where successful) and SP gives a measure of average number of iterations needed to achieve the threshold, adjusted in order to penalise algorithms with low success rates. *SRQEA* outperformed *RCQIEA* for all functions and for all thresholds. Function evaluations were kept to 300000.

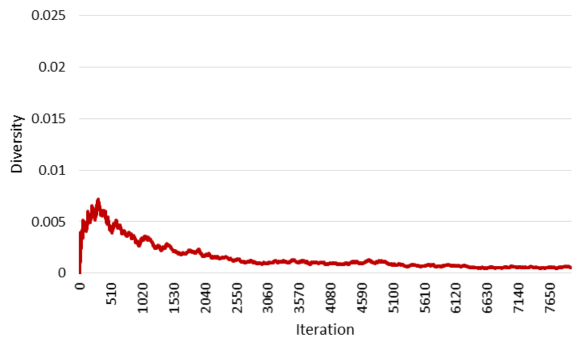
30 dimensions	RCQIEA								SRQEA							
	1.00E-02		1.00E-04		1.00E-06		1.00E-08		1.00E-02		1.00E-04		1.00E-06		1.00E-08	
Function name	SR	SP	SR	SP	SR	SP	SR	SP	SR	SP	SR	SP	SR	SP	SR	SP
Sphere	1.00	3.52E+05	0.02	5.91E+07	0.00	-	0.00	-	1.00	5.65E+04	1.00	1.04E+05	1.00	1.65E+05	1.00	2.48E+05
Schwefel 222	0.00	-	0.00	-	0.00	-	0.00	-	1.00	1.27E+05	1.00	2.82E+05	1.00	4.79E+05	1.00	7.19E+05
Schwefel 12	0.00	-	0.00	-	0.00	-	0.00	-	1.00	1.12E+05	1.00	1.75E+05	1.00	2.60E+05	1.00	3.56E+05
Step	1.00	7.77E+04	1.00	7.77E+04	1.00	7.77E+04	1.00	7.77E+04	1.00	4.27E+04	1.00	4.27E+04	1.00	4.27E+04	1.00	4.27E+04
Quartic	1.00	1.37E+05	1.00	1.37E+05	1.00	1.37E+05	1.00	1.37E+05	1.00	4.34E+04	1.00	4.35E+04	1.00	4.35E+04	1.00	4.35E+04
Schwefel 226	1.00	1.78E+05	1.00	8.54E+05	0.00	-	0.00	-	1.00	4.28E+04	1.00	7.94E+04	1.00	1.36E+05	1.00	2.12E+05
Basic Rastrigin	1.00	2.83E+05	0.18	6.57E+06	0.00	-	0.00	-	1.00	1.05E+05	1.00	1.28E+05	1.00	1.78E+05	1.00	2.53E+05
Basic Ackley	0.06	1.95E+07	0.00	-	0.00	-	0.00	-	0.94	5.66E+05	0.92	7.12E+05	0.88	9.39E+05	0.63	1.54E+06
Basic Griewank	0.53	1.07E+06	0.00	-	0.00	-	0.00	-	0.43	1.83E+05	0.31	3.83E+05	0.31	5.82E+05	0.31	8.50E+05
Penalised 1	1.00	5.61E+04	1.00	5.61E+04	1.00	5.61E+04	1.00	5.61E+04	1.00	3.85E+04	1.00	3.85E+04	1.00	3.85E+04	1.00	3.85E+04
Penalised 2	1.00	3.85E+04	1.00	3.85E+04	1.00	3.85E+04	1.00	3.85E+04	1.00	3.30E+04	1.00	3.30E+04	1.00	3.30E+04	1.00	3.30E+04
Diff. powers	0.98	5.25E+05	0.00	-	0.00	-	0.00	-	1.00	6.47E+04	1.00	1.37E+05	1.00	2.52E+05	1.00	3.80E+05
Rastrigin	1.00	4.41E+05	0.00	-	0.00	-	0.00	-	1.00	1.66E+05	1.00	1.87E+05	1.00	2.33E+05	1.00	3.01E+05
Lun. bi-Rastrigin	0.00	-	0.00	-	0.00	-	0.00	-	0.96	4.48E+05	0.94	5.17E+05	0.94	5.54E+05	0.94	6.09E+05



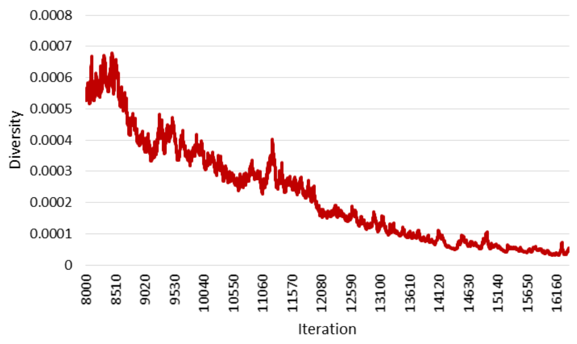
a)



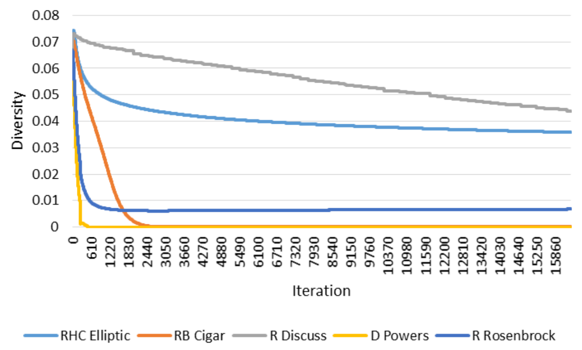
b)



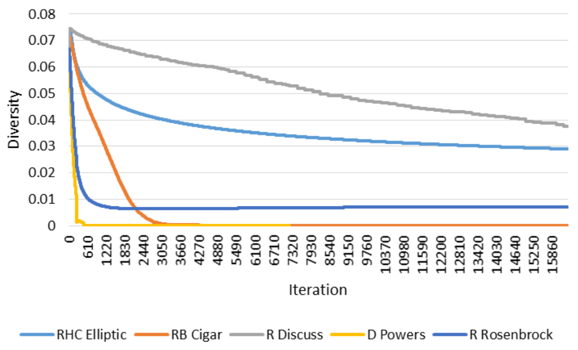
c)



d)



e)



f)

Fig. 8: Diversity over time for a) *Classic*, b) *HSB*, c) *RCQIEA* and d) *SRQEA*. For *Classic* and *HSB* minimum and mean diversity have been given for *Sphere* which is representative of all the results, with a scale up to 0.025. For *RCQIEA* and *SRQEA* mean diversity is given for *RHC Elliptic*, *RB Cigar*, *R Discuss*, *D Powers* and *R Rosenbrock*, with a scale up to 0.08, showing representative results for different patterns of behavior for these two algorithms.

Diversity over the timeline of evolution is shown in Fig. 8 for the four *QIEA*. *Classic* and *HSB* had consistent patterns of diversity. The restrictions placed on evolution of the *Qbits* by *HSB* results in a slower development of diversity, rising to a much lower peak than for *Classic*. However, a slight amount of noise is present in the tail of *HSB* which is lacking in Classic. This noise is due to the combination of the stochastic nature of *QIEA*, and the softer limiting conditions of *HSB* resulting from the slower evolution of the LSBs. However, even for both binary algorithms the diversity reduces rapidly, suggesting that improvements may be made by directly addressing this issue (for example mutation operators or immigration schemes).

The two real QIEA had different patterns for different functions, although were consistent relative to each other. Some functions, such as *Rotated Bent Cigar* and *Different Powers* rapidly reduced their diversity, while others had a slower decline. The quicker reductions in diversity do correlate with the better performances of the real QIEA from Table 3. Nevertheless, it may be beneficial to incorporate some threshold detection of low diversity and add some in through mutation or immigration, for example. A detection method is needed since the variety of evolution patterns seen in the diversity timelines indicates a need for adaptable methods, to keep diversity evolution consistent across different optimisation tasks.

5.3. Comparison of QIEA with published results

As the best performing *QIEA* on the traditional test functions, we compare *SRQEA* to two other algorithms – *FEP* [12] and *MADE* [45] (Table 6). Comparison is made difficult by varying numbers of function evaluations across the published methods, but in general, *SRQEA* outperformed *FEP* except for the *Rosenbrock*, *Ackley* and *Griewank* functions where *FEP* had a superior mean and standard deviation. *MADE* was better than *SRQEA* for *Schwefel-2.21*, *Rosenbrock*, *Ackley* and *Griewank*, but *SRQEA* beat *MADE* for *Quartic* and matched it for all of the other functions. Unfortunately, best minimum values found were not published for either algorithm, but since *MADE* produced several zero means, it is clear those results would have been good as well.

Table 6: Comparison between *SRQEA*, *Fast Evolutionary Programming* (*FEP*) [12], and *MADE* [45] on the traditional test functions. The *SRQEA* performed better than *FEP*, but was inferior to *MADE* for four of the functions. Best values are highlighted in bold type.

30 Dimensions	*SRQEA*				*FEP*			*MADE*		
Function	Func Evals	Min	Mean	Std dev	Func Evals	Mean	Std dev	Func Evals	Mean	Std dev
1 Sphere	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	8.10E-03	7.70E-04	150000	**0.00E+00**	**0.00E+00**
2 Schwefel-2.22	300000	0.00E+00	**0.00E+00**	**0.00E+00**	200000	8.10E-03	7.70E-04	150000	**0.00E+00**	**0.00E+00**
3 Schwefel-1.2	300000	0.00E+00	**0.00E+00**	**0.00E+00**	500000	1.60E-02	1.40E-02	200000	**0.00E+00**	**0.00E+00**
4 Schwefel-2.21	300000	3.51E-03	6.16E-03	1.56E-03	500000	3.00E-01	5.00E-01	500000	**0.00E+00**	**0.00E+00**
5 Rosenbrock	300000	1.04E-02	8.86E+01	1.80E+02	2000000	5.06E+00	5.87E+00	500000	**3.97E-01**	**1.63E+00**
6 Step	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	**0.00E+00**	**0.00E+00**	500000	**0.00E+00**	**0.00E+00**
7 Quartic	300000	0.00E+00	**0.00E+00**	**0.00E+00**	300000	7.60E-03	2.60E-03	300000	1.24E-03	3.78E-04
8 Schwefel-2.26	300000	0.00E+00	**0.00E+00**	**0.00E+00**	900000	1.50E+01	5.26E+01	200000	**0.00E+00**	**0.00E+00**
9 Basic Rastrigin	300000	0.00E+00	**0.00E+00**	**0.00E+00**	500000	4.60E-02	1.20E-02	300000	**0.00E+00**	**0.00E+00**
10 Basic Ackley	300000	0.00E+00	9.20E-01	4.01E+00	150000	1.80E-02	2.10E-03	150000	**0.00E+00**	**0.00E+00**
11 Basic Griewank	300000	0.00E+00	2.06E-02	2.25E-02	200000	1.60E-02	2.20E-02	200000	**0.00E+00**	**0.00E+00**
12 Penalised-1	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	9.20E-06	3.60E-06	300000	**0.00E+00**	**0.00E+00**
13 Penalised-2	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	1.60E-04	7.30E-05	300000	**0.00E+00**	**0.00E+00**

The exploitation abilities of *RCQIEA* and *SRQEA* were compared to data published on a set of differential algorithms (*DE*) [45]. The results are presented in Table 7, using the success rate (*SR*) and success performance (*SP*) metrics. In general, the *DE* algorithms achieved success more often, and quicker than the *rQIEA*. The *SRQEA* is superior to *RCQIEA* for these metrics, achieving better success rates, and reaching the threshold more quickly (better *SP*). These results (Table 7) represent the weakest performance for the *QIEA* in this paper, and indicate room for improvement in their search and exploitation abilities for the traditional test functions. However, it should be noted that success rates were based on very low thresholds (usually 1e-08) and therefore may not be important in practical cases. Unfortunately, *MADE* was not applied to the *CEC-2013* functions, so we cannot argue if these conclusions hold for the more complicated test functions.

Table 7: Comparison of success rates (SR) and speed of convergence (SP), between *RCQIEA*, *SRQEA* and 4 differential evolution algorithms, for the 13 traditional test functions with 30 dimensions. The threshold (1E-08, except of 1E-02 for *Quartic*) determines the point at which a run is a success. Best values are highlighted in bold type. Function evaluations are kept to 300000.

Function	*RCQIEA*		*SRQEA*		*jDE*		*SDE*		*JADE*		*MADE*	
	SP	*SR*	*SP*	*SR*	*SP*	*SR*	*SP*	*SR*	*SP*	*SR*	*SP*	*SR*
1 Sphere	—	0	2.48E+05	**1**	5.93E+04	**1**	3.91E+04	**1**	3.04E+04	**1**	**2.29E+04**	**1**
2 Schwefel-2.22	—	0	7.19E+05	**1**	8.16E+04	**1**	5.31E+04	**1**	5.61E+04	**1**	**3.64E+04**	**1**
3 Schwefel-1.2	—	0	3.56E+05	**1**	3.37E+05	**1**	—	0	**7.17E+04**	**1**	1.34E+05	**1**
4 Schwefel-2.21	—	0	—	0	2.99E+05	**1**	4.72E+05	0.44	—	0	**1.27E+05**	**1**
5 Rosenbrock	—	0	—	0	5.89E+06	0.08	—	0	**1.22E+05**	**0.92**	1.97E+05	**0.92**
6 Step	7.77E+04	**1**	4.27E+04	**1**	2.27E+04	**1**	1.44E+04	**1**	1.16E+04	**1**	**7.89E+03**	**1**
7 Quartic	1.37E+05	**1**	4.35E+04	**1**	1.12E+05	**1**	8.34E+04	**1**	2.97E+04	**1**	**2.83E+04**	**1**
8 Schwefel-2.26	—	0	2.12E+05	**1**	7.85E+04	**1**	**5.50E+04**	**1**	1.00E+05	**1**	6.00E+04	**1**
9 Basic Rastrigin	—	0	2.53E+05	**1**	1.17E+05	**1**	6.14E+05	0.36	1.31E+05	**1**	**1.14E+05**	**1**
10 Basic Ackley	—	0	1.54E+06	0.63	9.02E+04	**1**	5.95E+04	**1**	4.75E+04	**1**	**3.55E+04**	**1**
11 Basic Griewank	—	0	8.50E+05	0.31	6.21E+04	**1**	4.07E+04	**1**	3.30E+04	**1**	**2.41E+04**	**1**
12 Penalised-1	5.61E+04	**1**	3.85E+04	**1**	5.40E+04	**1**	3.66E+04	**1**	2.95E+04	**1**	**2.03E+04**	**1**
13 Penalised-2	3.85E+04	**1**	3.30E+04	**1**	5.76E+04	**1**	3.77E+04	**1**	2.95E+04	**1**	**2.19E+04**	**1**

A comparison of *SRQEA* with five different *QIEAs* is given in Table 8: a hybrid quantum PSO algorithm *HRCQEA* [37], a region based *QIEA RQEA* [42], a hybrid quantum PSO with neighbourhood search *NQPSO* [43], and two hybrid quantum GAs *QGAXM* [54] and *CQGA* [44]. The five fitness functions used in [37] where available in [42] and [43], so were chosen for comparison. When comparing to *QGAXM* and *CQGA*, the evaluated fitness functions were matched in their entirety, including a two-dimensional problem from [44]. The number of runs and the maximum function evaluations were matched, except for *HRCQEA*, where our algorithms performed only 300000 evaluations. It can be concluded that *SRQEA* is as good as, or better than these algorithms for finding the functions' minimum values, with the exception when against *QGAXM,* where *SRQEA* was better for the multi-modal problems, but worse for *Sphere* and *Rosenbrock*. Mean performance was less impressive, and suggests a weakness in exploitation capabilities of *SRQEA* for the basic functions, especially when using a low number of function evaluations when compared to *QGAXM*. In the next section though, evidence of a very good exploration for the more complicated *CEC-2013* functions will be presented. The *CQGA* algorithm used binary encoding, but with only 20 bits, and was beaten not just by *SRQEA*, but by *HSB* as well. Otherwise, *HSB* only achieved superior performance for *Rastrigin* against *QGAXM* and *SRQEA*.

Table 8: Comparison of *HSB* and *SRQEA* with five *QIEA: HRCQEA*; *RQEA*; *NQPSO; QGAXM; and CQGA*. Comparisons to *HRCQEA*, *RQEA* and *NQPSO* were standardised to the five functions used for *HRCQEA* [37], whereas for *QGAXM* and *CQGA* comparisons were made for all fitness functions presented. Values less than 1e-08 have been clamped to zero. Minimum scores for the compared algorithms are listed where available or where they can be deduced from zero means. Number of runs and function evaluations (FE) have been matched, except for *HRCQEA* (*) where only 300000 evaluations were performed per run. Best minimums are highlighted in bold except for the *CQGA* comparison which was a maximisation problem.

	Compared algorithm				HSB		SRQEA	
Method	Func	Dim	Min	Mean	Min	Mean	Min	Mean
HRCQEA 50 runs *2400000 FE	Sphere	30	**0.00E+00**	0.00E+00	1.40E+01	6.37E+02	**0.00E+00**	0.00E+00
	Rastrigin	30	**0.00E+00**	0.00E+00	1.62E+01	4.15E+01	**0.00E+00**	0.00E+00
	Ackley	30	1.70E-07	1.70E-07	1.02E+01	1.66E+01	**0.00E+00**	9.20E-01
	Schwefel 7	30	3.90E-04	3.90E-04	2.91E+02	7.10E+02	**0.00E+00**	2.60E+02
	Griewank	30	**0.00E+00**	0.00E+00	1.83E+00	7.69E+00	**0.00E+00**	2.06E-02
RQEA 25 runs 500000 FE	Sphere	50	**0.00E+00**	0.00E+00	3.80E+02	1.65E+03	**0.00E+00**	0.00E+00
	Rastrigin	50	-	5.32E-07	4.61E+01	6.36E+01	**0.00E+00**	0.00E+00
	Ackley	50	**0.00E+00**	0.00E+00	1.09E+01	1.72E+01	**0.00E+00**	8.36E-08
	Schwefel 7	50	-	5.80E-03	5.84E+02	1.64E+03	**0.00E+00**	6.40E+02
	Griewank	50	**0.00E+00**	0.00E+00	4.78E+00	1.77E+01	**0.00E+00**	2.36E-02
NQPSO 30 runs 200000 FE	Sphere	30	**0.00E+00**	0.00E+00	9.37E+01	5.79E+02	**0.00E+00**	0.00E+00
	Rastrigin	30	**0.00E+00**	0.00E+00	2.07E+01	3.54E+01	**0.00E+00**	0.00E+00
	Ackley	30	**0.00E+00**	0.00E+00	1.57E+01	1.94E+01	1.64E-08	1.49E+00
	Schwefel 7	30	-	3.80E+03	4.46E+02	8.17E+02	**0.00E+00**	2.89E+02
	Griewank	30	**0.00E+00**	0.00E+00	2.13E+00	6.90E+00	**0.00E+00**	2.54E-02
QGAXM 30 runs 10000 FE	Sphere	50	**1.90E-01**	4.20E-01	7.80E+04	9.43E+04	1.40E+01	8.43E+01
	Rastrigin	50	1.67E+04	1.35E+05	**6.41E+02**	**6.92E+02**	1.67E+05	2.09E+06
	Rosenbrock	50	3.20E+02	4.61E+02	2.19E+10	4.50E+10	**1.87E+01**	2.86E+01
	Griewank	50	1.44E+00	2.22E+00	6.66E+02	8.56E+02	**1.12E+00**	1.58E+00
CQGA, 10 runs, 8000 FE	Complex binary	2	-17.3503	-	**-17.4503**	-17.4486	**-17.4503**	-17.4503

Table 9 shows the performance of *SRQEA* against two algorithms that were applied to the *CEC-2013* fitness functions [13]. The two algorithms compared are a particle swarm optimization algorithm *SPSO-2011* [15] and a genetic algorithm *GA* [17]. *SRQEA* was chosen for comparison as, overall, it was the best performing *QIEA* tested here, in terms of minimum values found.

Table 9: *SRQEA* compared to *SPSO-2011* and a GA algorithm for the *CEC-2013* functions with 50 dimensions and 500000 FEs. The *SRQEA* matched or outperformed the other two algorithms on best value found (Min) for 11 test functions. Best values are highlighted in bold type.

50 Dimensions	*SRQEA*			*SPSO-2011* [15]			*GA* [17]			
Function	Min	Mean	Std dev	Min	Median	Std dev	Min	Median	Mean	Std dev
14 Sphere [duplicated]	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.18E-13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	1.55E+06	2.96E+06	6.43E+05	3.79E+05	6.80E+05	**1.87E+05**	**1.74E+05**	**4.28E+05**	**4.76E+05**	2.14E+05
16 Rotated bent cigar	**4.98E-02**	**1.58E+05**	**1.08E+06**	2.00E+07	4.37E+08	9.47E+08	2.55E+06	**3.44E+07**	1.06E+08	1.49E+08
17 Rotated discus	1.14E+05	1.75E+05	2.47E+04	3.22E+04	5.10E+04	8.72E+03	**4.90E-01**	**2.25E+00**	**3.33E+00**	**4.88E+00**
18 Different powers	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.41E-05	**0.00E+00**	**0.00E+00**	4.77E+04	1.70E+05
19 Rotated Rosenbrock	2.38E+01	**4.19E+01**	**7.54E+00**	**1.84E+01**	**4.35E+01**	2.41E+01	3.66E+01	4.36E+01	4.72E+01	1.40E+01
20 Rotated Schaffers-F7	1.47E+02	2.46E+02	9.28E+01	5.61E+01	8.64E+01	**1.53E+01**	**1.51E+01**	**3.97E+01**	**4.17E+01**	1.83E+01
21 Rotated Ackley	**2.10E+01**	**2.11E+01**	4.68E-02	2.10E+01	**2.11E+01**	4.25E-02	2.11E+01	2.12E+01	2.12E+01	**3.98E-02**
22 Rotated Weierstrass	6.16E+01	**6.74E+01**	**3.76E+00**	**4.52E+01**	**5.40E+01**	6.74E+00	5.21E+01	7.53E+01	7.43E+01	3.97E+00
23 Rotated Griewank	**2.71E-02**	1.31E-01	**4.96E-02**	1.00E-01	4.00E-01	2.38E-01	2.71E-02	**9.36E-02**	**1.05E-01**	7.09E-02
24 Rastrigin	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.50E+02	2.30E+02	4.18E+01	1.49E+01	**5.37E+01**	5.57E+01	2.23E+01
25 Rotated Rastrigin	4.60E+02	6.85E+02	1.38E+02	1.62E+02	2.35E+02	4.87E+01	**5.07E+01**	**9.75E+01**	**9.83E+01**	**2.45E+01**
26 NC rotated Rastrigin	5.01E+02	6.89E+02	1.01E+02	3.20E+02	4.28E+02	6.22E+01	**1.04E+02**	**1.86E+02**	**1.93E+02**	**5.30E+01**
27 Schwefel-7	**9.99E-02**	**6.71E-01**	**2.94E-01**	5.51E+03	7.26E+03	8.53E+02	1.06E+03	**2.30E+03**	2.55E+03	1.14E+03
28 Rotated Schwefel-7	**4.69E+03**	**6.22E+03**	**6.23E+02**	5.68E+03	**7.92E+03**	1.14E+03	6.20E+03	8.24E+03	9.84E+03	3.19E+03
29 Rotated Katsuura	**8.93E-01**	**1.83E+00**	4.41E-01	1.40E+00	**2.00E+00**	**3.87E-01**	2.23E+00	3.76E+00	3.68E+00	3.88E-01
30 Lunacek bi-Rastrigin	**0.00E+00**	**1.96E-04**	**1.40E-03**	2.08E+02	3.11E+02	6.62E+01	8.25E+01	**1.13E+02**	1.15E+02	2.00E+01
31 R Lunacek bi-Rastrigin	4.53E+02	6.12E+02	9.30E+01	1.70E+02	2.91E+02	**6.24E+01**	**8.83E+01**	**1.32E+02**	**1.68E+02**	1.02E+02
32 RE Griewank Rosenb.	1.46E+02	2.91E+02	6.26E+01	1.70E+01	3.72E+01	1.20E+01	**3.60E+00**	**9.02E+00**	**8.92E+00**	**3.17E+00**
33 RE Schaffers-F6	**1.90E+01**	2.44E+01	**7.68E-01**	1.99E+01	**2.27E+01**	1.19E+00	1.99E+01	2.36E+01	**2.35E+01**	8.02E-01

Looking at all dimensions, all three algorithms achieved some best performances. However, *SPSO-2011* performed least well, having fewer best minimum results, and most of those being joint equal with one or both of the other algorithms. The main competition for *SRQEA* came from the *GA*. For 10 dimensions it achieved 16 best performances, with *SRQEA* only achieving seven. For 30 dimensions *GA* scored 12 best performances, while the *SRQEA* reached 8, but for 50 dimensions, *SRQEA* took the lead with 11 compared to 9 best results for the *GA*. This demonstrates better scaling with increased number of dimensions for *SRQEA* than for the *GA*. Mean performance was similarly distributed across all dimensions but *SRQEA* showed improved standard deviation performance again for 50 dimensions, outperforming the other algorithms substantially. This shows a more consistent relative performance at higher dimensions for *SRQEA* as well as better minima and means.

The poorer performance of *SPSO-2011* (Table 9) and the better performance of the *GA* may suggest that the recombinatorial properties of the cross-over operator may aid the search pattern for the *CEC-2013* functions. This may be consistent with either of our hypotheses for why the *bQIEA* performed relatively well against the *rQIEA* – either treating the rougher space as more discrete and looking for recombination, or navigating through hops (swapping genes in the case of *GA*, and flipping bits in the case of the *bQIEA*). Although overall *SRQEA* was better, it would be interesting to see how *bQIEA* perform against *rQIEA* and other algorithms on even more complex search spaces.

A comparison between *SRQEA* and two alternative algorithms, when applied to the real-world problems is shown in Table 10. For the *frequency modulation wave matching problem*, *MADE-WS* [45] had the best mean and standard deviation. Unfortunately, the authors did not report a minimum value. *SRQEA* outperformed the hybrid algorithm [55] and *the DE* algorithm [56], in terms of mean and standard deviation, while equalling the best minimum performance. The mean and standard deviation were worse but comparable with the *MADE-WS* results.

Table 10: Comparison of performance on real-world problems between *SRQEA* and three alternative algorithms – *MADE-WS* [45], *EA-DE-Memetic* [55] and an adaptive differential evolutionary algorithm [56]. The starred value has been clamped to zero as it was below the threshold of 1E-08 (used in our simulations). Best values are highlighted in bold type. Function evaluations are kept to 150000.

Function	SRQEA			MADE-WS			EA-DE-Memetic			Adaptive DE		
	Min	Mean	Std dev	Min	Mean	Std dev	Min	Mean	Std dev	Min	Mean	Std dev
FM	**0.00E+00**	1.70E+01	4.68E+00	-	**8.81E-01**	**2.47E+00**	**0.00E+00***	3.81E+00	5.21E+00	**0.00E+00**	4.85E+00	6.69E+00
L-J 5	**-1.27E+01**	**-1.21E+01**	1.02E+00	-	-9.09E+00	**8.83E-02**	-	-	-	-	-	-
L-J 10	**-3.18E+01**	-2.41E+01	4.23E+00	-	-2.66E+01	**8.64E-01**	-2.84E+01	-2.59E+01	2.24E+00	-2.80E+01	**-2.68E+01**	2.11E+00
Radar	**1.59E+00**	**2.10E+00**	2.09E-01	-	-	-	2.20E+02	2.20E+02	**0.00E+00**	2.20E+02	2.20E+02	**0.00E+00**

For the *Lennard-Jones* problems, *SRQEA* again established the best minimum values, but *MADE-WS* did not have a comparable values published. *SRQEA* did have the best mean value for *Lennard-Jones5* but only outperformed the hybrid algorithm for *Lennard-Jones10*.

For the *radar waveform parameter specification problem*, *SRQEA* was the clear winner. The published results [55] and [56] both gave a suspiciously poor value though, and it may be worth considering whether there were issues in using shared code for the function evaluations. The problem was directly tackled in [57] where a *variable neighbourhood search algorithm* gave a minimum value of 8.58e-01 which was better than that achieved by the *SRQEA*.

Computational overhead is presented in Table 11. They were obtained using Microsoft Visual Studio Profiler, with a set of small runs for each algorithm, to achieve a representative sample, albeit subject to considerable noise. The orders of magnitude are representative however, and the scores are normalized so that 1.00 is the time taken to evaluate 1000 *sphere* fitness functions on 30 dimensions. *Classic* and *HSB* were substantially more computationally expensive per function evaluation and generation than the real value algorithms. By inspecting the code, we can conclude that this was due to substantially more trigonometric evaluations, one for each bit, compared to a small number used for the rotation gate in the real algorithms. Even so, the real algorithms also added considerable overhead. Replacing the trigonometric functions with approximations or look-up tables, reworking the algorithms to use more linear movements instead of rotations, or using hardware with fast *sin* and *cos* functions would greatly speed up the performance of the presented *QIEA*. The *sphere* function is very simple and quick to compute. However, the algorithms will add substantial overhead unless the fitness functions are very demanding.

Table 11: Algorithm computational overhead using Microsoft Visual Studio Profiler. Mean generation and FE overheads are presented, normalized so that 1.00 = time of 1000 *sphere* function evaluations in 30 dimensions. Values were subject to considerable noise but the orders of magnitude are accurate.

	Classic	HSB	RCQIEA	SRQEA
Generation overhead	8.93	8.47	0.71	0.21
Overhead per FE	878.89	816.06	0.95	1.07

6. Conclusion

When applied to real-value optimization tasks, all of the *QIEA* tested and validated in this investigation were successful, in that they were able to produce acceptable to excellent error values (with respect to the complexity of the test functions). Binary *QIEA* are a direct implementation of the quantum computing metaphor, which is built around repeated sampling of binary strings, analogous to superposition of states on a set of quantum bits. The Qbit probabilities define a probability distribution that elegantly specifies both the region of the best solution found so far, and the variance of the search radius. As the probabilities saturate, the mean position of the search becomes clearly defined, and the variance of the search narrows.

Although the *bQIEA* algorithms performed relatively poorly on some of the optimization tasks investigated here, they demonstrated good results for some of the *CEC-2013* test functions, even outperforming the rQIEA for some at the highest dimensions. In general, our modification (*HSB*) provided better results than *Classic*. The timeline and diversity plots highlighted the premature convergence of *Classic* (Fig. 7a), giving further justification for our choice of modification, which was developed in response to our analysis of individual bit evolution. By explicitly limiting the saturation of less significant bits to the magnitude of saturation of more significant bits, *HSB* avoids the issues that *Classic* encountered for real-value problems, although zero gradients in the latter half of some timeline plots suggest there is still room for improvement. The population size results (Fig. 4a and Fig. 4b) also suggest exploration issues, as the *bQIEA* benefit from a larger population size for a fixed number of function evaluations.

The best results came from the *rQIEA*, especially from our modified version - *SRQEA* (Table 2, Table 3 and Table 4). However, the *rQIEA* specifications require a compromise with respect to the quantum metaphor. In most *rQIEA*, and certainly the ones investigated here, the Qbits and the quantum rotation gate give a mechanism for adjusting the radius of search throughout the evolution, such as through the creep mutation operators of the *rQIEA* presented here, or in the velocity equations in PSO algorithms [38]. Although that is not a problem of itself, it may be useful to view these quantum inspired algorithms as operator algorithms used within other optimization methods. Presented on their own, *rQIEA* can largely resemble other techniques. For example, the *RCQIEA* algorithm used in this work looks similar to simulated annealing, with the rotation gate adjusting the variance for neighbour selection.

Our modification to the rotation gate produced superior results, particularly with regards to the final ability to exploit the search space (Table 2 and Table 3) and the speed of exploitation (Table 5), although from the heatmap of Fig. 5 it would appear the average performance across the functions is slightly compromised. This suggests the superior exploitation may come at the expense of some exploration capability. As well as being beneficial in this specific implementation, it would be interesting for future work to explore the possibility of using the modified rotation in other algorithms, as a way of adjusting search variance.

When compared to other published results, our modified algorithms were predominantly competitive for the more complex *CEC-2013* functions (Table 9). For the traditional test functions, *SRQEA* was superior than recently published *QIEA* in terms of best minimum reached (Table 8), although mean performance was mixed, and our algorithms were generally outperformed by other published results (in particular, the *DE* algorithms [45], Table 6 and Table 7). However, timeline plots (Fig. 7) suggest the *rQIEA* may continue to improve if left for longer. It would therefore be interesting to see if these algorithms are suitable for increasingly complicated test functions, where longer processing times are to be expected. Both *HSB* and *SRQEA* outperformed a *bQIEA* applied to a real value problem (Table 8).

Surprisingly, the *bQIEA* appeared to perform better for the more complex *CEC-2013* and the real-world test functions (Table 2, Table 3 and Table 4). We have speculated that this may be because either the transferred search space begins to resemble the binary space portioning that the *bQIEA* generate, or that the search hops at different scales (depending on bit significance) may result in more suitable search patterns when compared to *rQIEA* or other algorithms. The ability of *bQIEA* to combine different scales, through bit manipulation, may explain their improved performance on these more sophisticated tasks. As more complex fitness functions are published in the future, it would be worth including *bQIEA* (and perhaps other binary optimisation algorithms) in attempting to optimise them. All of the algorithms appeared to scale well to 100 dimensions, with the *bQIEA* performing well on some functions. A lack of published data on the benchmarks at 100 dimensions for other algorithms limits the conclusions that can be made, but the results had similar magnitudes of minimum fitness values compared to the 50 dimension results.

QIEA, and *rQIEA* in particular, provide a good starting point for optimization. Deficiencies, when compared to competing algorithms, were largely down to fine exploitation, with results being of a similar degree of magnitude in error (Table 6). Future work would be beneficial on improving exploration for *SRQEA*, or further reducing the premature convergence for *HSB*. This may be achieved through an analysis of the effect of changing algorithm parameters (as discussed below), or by including the *QIEA* in hybrid algorithms with a two-stage exploration and exploitation process. Using the configuration of step size and other parameters presented here, the two *rQIEA* are more orientated towards exploration than exploitation. This is demonstrated by the populations analysis (Fig. 4), which showed they both benefitted from a small population size for a given number of function evaluations (thereby increasing the number of iterations per individual). The *bQIEA* in contrast performed best with a larger population size and so appear to be balanced more towards exploitation than exploration.

One final advantage of *QIEA* is the low number of parameters they require for the main part of their implementation. Generally, only the number of individuals and step size for the rotation gate are needed. The *rQIEA* investigated in this work also include a parameter for the number of children produced in each generation. The step size investigation presented in Table 1 produced mixed results. Although we decided to keep a step size of π/100 for the bQIEA, in keeping with published work from other authors, our results suggest a slightly larger step size would be beneficial for future work. No clear conclusions could be made, however, for the *rQIEA* step size. The *bQIEA* also have parameters for local and global update rates, while *rQIEA* have crossover rates. How these affect the overall performance was not evaluated. The *rQIEA* also add a parameter for the number of offspring spawned at each iteration. Again, changing this was not analysed and further investigation into the optimisation of these parameters would be worth conducting.

References

[1] M. Kociecki, H. Adeli, Two-phase genetic algorithm for size optimization of free-form steel space-frame roof structures, J. Constr. Steel Res. 90 (2013) 283–296. doi:10.1016/j.jcsr.2013.07.027.

[2] M. Kociecki, H. Adeli, Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures, Eng. Appl. Artif. Intell. 32 (2014) 218–227. doi:10.1016/j.engappai.2014.01.010.

[3] B.R. Campomanes-Álvarez, O. Cordón, S. Damas, Evolutionary Multi-objective Optimization for Mesh Simplification of 3D Open Models, Integr Comput-Aided Eng. 20 (2013) 375–390. doi:10.3233/ICA-130443.

[4] T. Chabuk, J. Reggia, J. Lohn, D. Linden, Causally-guided evolutionary optimization and its application to antenna array design, Integr. Comput.-Aided Eng. 19 (2012) 111–124. doi:10.3233/ICA-2012-0395.

[5] M. Molina-García, J. Calle-Sánchez, C. González-Merino, A. Fernández-Durán, J.I. Alonso, Design of In-building Wireless Networks Deployments Using Evolutionary Algorithms, Integr Comput-Aided Eng. 21 (2014) 367–385. doi:10.3233/ICA-140474.

[6] M.R. AlRashidi, M.E. El-Hawary, A Survey of Particle Swarm Optimization Applications in Electric Power Systems, IEEE Trans. Evol. Comput. 13 (2009) 913–918. doi:10.1109/TEVC.2006.880326.

[7] C. Kyriklidis, G. Dounias, Evolutionary computation for resource leveling optimization in project management, Integr. Comput.-Aided Eng. (2015) 1–12.

[8] H. Tao, J.M. Zain, M.M. Ahmed, A.N. Abdalla, W. Jing, A wavelet-based particle swarm optimization algorithm for digital image watermarking, Integr. Comput.-Aided Eng. 19 (2012) 81–91. doi:10.3233/ICA-2012-0392.

[9] W.-Y. Hsu, Application of quantum-behaved particle swarm optimization to motor imagery EEG classification, Int. J. Neural Syst. 23 (2013) 1350026. doi:10.1142/S0129065713500263.

[10] S. Das, P. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur Univ. Nanyang Technol. Univ. Kolkata. (2010).

[11] M. Martínez-Ballesteros, J. Bacardit, A. Troncoso, J.C. Riquelme, Enhancing the Scalability of a Genetic Algorithm to Discover Quantitative Association Rules in Large-scale Datasets, Integr Comput-Aided Eng. 22 (2015) 21–39. doi:10.3233/ICA-140479.

[12] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102. doi:10.1109/4235.771163.

[13] B.Y.Q. J. J.Liang, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Tech. Rep. 201212 Comput. Intell. Lab. Zhengzhou Univ. Zhengzhou China. (2013).

[14] M. Reyes-Sierra, C.C. Coello, Multi-objective particle swarm optimizers: A survey of the state-of-the-art, Int. J. Comput. Intell. Res. 2 (2006) 287–308.

[15] M. Zambrano-Bigiarini, M. Clerc, R. Rojas, Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements, in: 2013 IEEE Congr. Evol. Comput. CEC, 2013: pp. 2337–2344. doi:10.1109/CEC.2013.6557848.

[16] M. Srinivas, L.M. Patnaik, Genetic algorithms: a survey, Computer. 27 (1994) 17–26. doi:10.1109/2.294849.

[17] S.M. Elsayed, R.A. Sarker, D.L. Essam, A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization, in: 2013 IEEE Congr. Evol. Comput. CEC, 2013: pp. 356–360. doi:10.1109/CEC.2013.6557591.

[18] S. Das, P.N. Suganthan, Differential Evolution: A Survey of the State-of-the-Art, IEEE Trans. Evol. Comput. 15 (2011) 4–31. doi:10.1109/TEVC.2010.2059031.

[19] J. Tvrdik, R. Polakova, Competitive differential evolution applied to CEC 2013 problems, in: 2013 IEEE Congr. Evol. Comput. CEC, 2013: pp. 1651–1657. doi:10.1109/CEC.2013.6557759.

[20] M.M. Joly, T. Verstraete, G. Paniagua, Integrated Multifidelity, Multidisciplinary Evolutionary Design Optimization of Counterrotating Compressors, Integr Comput-Aided Eng. 21 (2014) 249–261. doi:10.3233/ICA-140463.

[21] X.X. Li, W.D. Li, X.T. Cai, F.Z. He, A hybrid optimization approach for sustainable process planning and scheduling, Integr. Comput.-Aided Eng. 22 (2015) 311–326. doi:10.3233/ICA-150492.

[22] J.-F. Chen, T.-J. Wu, A Computational Intelligence Optimization Algorithm: Cloud Drops Algorithm, Integr Comput-Aided Eng. 21 (2014) 177–188. doi:10.3233/ICA-130459.

[23] J.P. Cunningham, Z. Ghahramani, Linear Dimensionality Reduction: Survey, Insights, and Generalizations, J. Mach. Learn. Res. 16 (2015) 2859–2900.

[24] J. Hua, H. Wang, M. Ren, H. Huang, Dimension reduction using collaborative representation reconstruction based projections, Neurocomputing. 193 (2016) 1–6. doi:10.1016/j.neucom.2016.01.060.

[25] IEEE Xplore Abstract - A Survey of Sparse Representation: Algorithms and Applications, (n.d.). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7102696 (accessed May 10, 2016).

[26] W. Yang, Z. Wang, C. Sun, A collaborative representation based projections method for feature extraction, Pattern Recognit. 48 (2015) 20–27. doi:10.1016/j.patcog.2014.07.009.

[27] D. Wang, H. Lu, M.-H. Yang, Kernel collaborative face recognition, Pattern Recognit. 48 (2015) 3025–3037. doi:10.1016/j.patcog.2015.01.012.

[28] B. Cheng, J. Yang, S. Yan, Y. Fu, T.S. Huang, Learning With -Graph for Image Analysis, IEEE Trans. Image Process. 19 (2010) 858–866. doi:10.1109/TIP.2009.2038764.

[29] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, Evol. Comput. IEEE Trans. On. 6 (2002) 580–593.

[30] M.D. Platel, S. Schliebs, N. Kasabov, Quantum-inspired evolutionary algorithm: a multimodel EDA, Evol. Comput. IEEE Trans. On. 13 (2009) 1218–1232.

[31] G.K. Venayagamoorthy, G. Singhal, Quantum-Inspired Evolutionary Algorithms and Binary Particle Swarm Optimization for Training MLP and SRN Neural Networks, J. Comput. Theor. Nanosci. 2 (2005) 561–568. doi:10.1166/jctn.2005.011.

[32] H. Xing, Y. Ji, L. Bai, X. Liu, Z. Qu, X. Wang, An adaptive-evolution-based quantum-inspired evolutionary algorithm for QoS multicasting in IP/DWDM networks, Comput. Commun. 32 (2009) 1086–1094. doi:10.1016/j.comcom.2008.12.036.

[33] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithms with a new termination criterion, Hε gate, and two-phase scheme, IEEE Trans. Evol. Comput. 8 (2004) 156–169. doi:10.1109/TEVC.2004.823467.

[34] G. Zhang, H. Rong, Real-Observation Quantum-Inspired Evolutionary Algorithm for a Class of Numerical Optimization Problems, in: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot (Eds.), Comput. Sci. – ICCS 2007, Springer Berlin Heidelberg, 2007: pp. 989–996. http://link.springer.com/chapter/10.1007/978-3-540-72590-9_151 (accessed September 24, 2014).

[35] Q. Chaoyong, L. Yongjuan, Z. Jianguo, A real-coded quantum-inspired evolutionary algorithm for global numerical optimization, in: 2008: pp. 1160–1164. doi:10.1109/ICCIS.2008.4670779.

[36] G.S. Babu, D.B. Das, C. Patvardhan, Real-parameter quantum evolutionary algorithm for economic load dispatch, Gener. Transm. Distrib. IET. 2 (2008) 22–31.

[37] M.A. Hossain, M.K. Hossain, M.M.A. Hashem, A Generalized Hybrid Real-Coded Quantum Evolutionary Algorithm Based on Particle Swarm Theory with Arithmetic Crossover, Int. J. Comput. Sci. Inf. Technol. 2 (2010) 172–187. doi:10.5121/ijcsit.2010.2415.

[38] J. Xiao, J. Xu, Z. Chen, K. Zhang, L. Pan, A hybrid quantum chaotic swarm evolutionary algorithm for DNA encoding, Comput. Math. Appl. 57 (2009) 1949–1958. doi:10.1016/j.camwa.2008.10.021.

[39] Z. Tu, Y. Lu, Corrections to “A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization,” Evol. Comput. IEEE Trans. On. 12 (2008) 781–781. doi:10.1109/TEVC.2008.926734.

[40] G. Zhang, Quantum-inspired evolutionary algorithms: a survey and empirical study, J. Heuristics. 17 (2011) 303–351.

[41] R. Zhang, H. Gao, Real-coded Quantum Evolutionary Algorithm for Complex Functions with High-dimension, in: Int. Conf. Mechatron. Autom. 2007 ICMA 2007, 2007: pp. 2974–2979. doi:10.1109/ICMA.2007.4304033.

[42] T.-C. Lu, J.-C. Juang, A region-based quantum evolutionary algorithm (RQEA) for global numerical optimization, J. Comput. Appl. Math. 239 (2013) 1–11. doi:10.1016/j.cam.2012.09.015.

[43] X. Fu, W. Liu, B. Zhang, H. Deng, Quantum Behaved Particle Swarm Optimization with Neighborhood Search for Numerical Optimization, Math. Probl. Eng. 2013, 2013 (2013) e469723. doi:10.1155/2013/469723, 10.1155/2013/469723.

[44] H. Wang, J. Liu, J. Zhi, C. Fu, The Improvement of Quantum Genetic Algorithm and Its Application on Function Optimization, Math. Probl. Eng. 2013 (2013) e730749. doi:10.1155/2013/730749.

[45] J. Cheng, G. Zhang, F. Caraffini, F. Neri, Multicriteria adaptive differential evolution for global numerical optimization, Integr. Comput.-Aided Eng. 22 (2015) 103–107. doi:10.3233/ICA-150481.

[46] Y. Chi, F. Sun, L. Jiang, C. Yu, An efficient population diversity measure for improved particle swarm optimization algorithm, in: 2012 6th IEEE Int. Conf. Intell. Syst., 2012: pp. 361–367. doi:10.1109/IS.2012.6335243.

[47] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Enhancing Differential Evolution Utilizing Proximity-Based Mutation Operators, IEEE Trans. Evol. Comput. 15 (2011) 99–119. doi:10.1109/TEVC.2010.2083670.

[48] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: 2013 IEEE Congr. Evol. Comput. CEC, 2013: pp. 1952–1959. doi:10.1109/CEC.2013.6557798.

[49] A. Zamuda, J. Brest, E. Mezura-Montes, Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies on CEC 2013 real parameter optimization, in: 2013 IEEE Congr. Evol. Comput. CEC, 2013: pp. 1925–1931. doi:10.1109/CEC.2013.6557794.

[50] J.L. Rueda, I. Erlich, Hybrid Mean-Variance Mapping Optimization for solving the IEEE-CEC 2013 competition problems, in: 2013 IEEE Congr. Evol. Comput. CEC, 2013: pp. 1664–1671. doi:10.1109/CEC.2013.6557761.

[51] J.L.F. Martínez, E.G. Gonzalo, The Generalized PSO: A New Door to PSO Evolution, J Artif Evol App. 2008 (2008) 5:1–5:15. doi:10.1155/2008/861275.

[52] V.K. Koumousis, C.P. Katsaras, A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance, IEEE Trans. Evol. Comput. 10 (2006) 19–28. doi:10.1109/TEVC.2005.860765.

[53] H.-B. Duan, C.-F. Xu, Z.-H. Xing, A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems, Int. J. Neural Syst. 20 (2010) 39–50. doi:10.1142/S012906571000222X.

[54] A.M. Mohammed, N.A. Elhefnawy, M.M. El-Sherbiny, M.M. Hadhoud, Quantum inspired evolutionary algorithms with parametric analysis, in: Sci. Inf. Conf. SAI 2014, 2014: pp. 280–290. doi:10.1109/SAI.2014.6918202.

[55] H.K. Singh, T. Ray, Performance of a hybrid EA-DE-memetic algorithm on CEC 2011 real world optimization problems, in: 2011 IEEE Congr. Evol. Comput. CEC, 2011: pp. 1322–1326. doi:10.1109/CEC.2011.5949769.

[56] M. Asafuddoula, T. Ray, R. Sarker, An adaptive differential evolution algorithm and its performance on real world optimization problems, in: 2011 IEEE Congr. Evol. Comput. CEC, 2011: pp. 1057–1062. doi:10.1109/CEC.2011.5949734.

[57] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, M. Čangalović, Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search, Eur. J. Oper. Res. 151 (2003) 389–399. doi:10.1016/S0377-2217(02)00833-0.

Ivan Jordanov received his PhD degree in computer aided optimization and MSc in applied mathematics and informatics from the Technical University of Sofia (Bulgaria). He is currently a Reader in Computational Intelligence at the School of Computing, University of Portsmouth, UK. His research interests include computational intelligence (neural networks, evolutionary computation, heuristic global optimisation and data analytics) for solving pattern recognition and classification problems. He is an associate editor of two international journals, co-author several textbooks, and his publication list includes more than 80 papers.

Joe Wright received his PhD from University of Portsmouth (UK) and his BSc (Hons) in applied mathematics and statistics from the Open University (UK). His research interests include computational intelligence (heuristic global optimisation, evolutionary computation and pattern generation methods) for sport optimisation problems.