

Robert Nowotniak<sup>1</sup>, Jacek Kucharski<sup>2</sup>

# Meta-optimization of Quantum-Inspired Evolutionary Algorithm

## 1. Introduction

In this paper, a meta-optimization algorithm, based on Local Unimodal Sampling (LUS), has been applied to tune selected parameters of Quantum-Inspired Evolutionary Algorithm for numerical optimization problems coded in real numbers. Tuning of the following two parameters has been considered: crossover rate and contraction factor. Performance landscapes of the algorithm meta-fitness have been approximated and numerical experiments have been conducted on four high-dimensional optimization problems. The tuned algorithm outperformed the algorithm with initially proposed parameters in the conducted experiments.

The *quantum-inspired evolutionary algorithms* are located in the intersection of two subfields of computer science: quantum computing and evolutionary computing. Quantum computing[34] is a new branch of computer science concerning applications of unique quantum mechanical effects to solving computational problems. By application of such unique quantum mechanical phenomena it is possible to solve selected computational problems extremely efficiently. Contrary to “true” quantum algorithms (e.g. Grover’s search algorithm[15] or Shor’s factorization algorithm[40]), the considered algorithms do not require a functional quantum computer for their efficient implementation. Instead, the algorithms exploit additional level of randomness inspired by concepts and principles drawn from quantum mechanical systems, such as *qubits* or *superposition of states*. The first proposal of evolutionary algorithm based on the concepts and principles of quantum computing was presented in [33] and this area is still intensively studied nowadays. Other early examples of quantum-inspired genetic algorithms which employ binary quantum representation based on qubits are due to Han and Kim[17,18,20]. In the past decade, several other variants [1,5,6,7,9,39] of quantum evolutionary algorithms with real numbers representation have been also proposed. To simplify the notation, the algorithms will be

---

<sup>1</sup> MSc, Computer Engineering Department, Technical University of Lodz, Lodz, Poland

<sup>2</sup> MSc, PhD, DSc, Computer Engineering Department, Technical University of Lodz, Lodz, Poland

also denoted henceforth in this paper as *quantum evolutionary algorithms*. Despite no real quantum-level hardware is required for their efficient implementation, a possibility of potential implementation of the algorithms on a quantum computer is quite probable in the future[43]. This paper concerns the Real-Coded Quantum-Inspired Evolutionary Algorithm (RCQiEA), which has been presented in [5].

Recent years have witness successful applications of quantum-inspired evolutionary algorithms in variety of different areas, including image processing[13,21,23,40,41,42], network design problems[26,45,46], flow shop scheduling problems[16,48], thermal unit commitment[22], power system optimization[44], discovering structures in time series[3] and others[8,28]. Quantum evolutionary algorithms with real numbers representation have been also successfully applied in various fields: engineering optimization problems[1], option pricing modelling [10,12], power system optimization[2], financial data analysis[9,11], training fuzzy neural networks [47] and ceramic grinding optimization[29]. Consequently, it has been demonstrated that quantum evolutionary algorithms are capable to outperform classical metaheuristics for a wide range of problems.

Though several successful attempts [17,18,19,25] have been made for some specific algorithms, no strong theoretical foundations of general quantum evolutionary algorithms exist or the models are highly oversimplified. Also, no systematic approach to tuning parameters of the algorithms has been taken yet. Therefore, no general rules for setting parameters of quantum evolutionary algorithms are currently identified and it is the main motivation behind this paper. Usually, the researcher's experience in setting the parameters is required. In [5], few hints have been given on setting the value of the RCQiEA algorithm parameters. Neither the influence of the parameter value on the efficacy of the algorithm has been investigated. The aim of this paper was to analyse influence of the algorithm parameters and to find their optimum values. A meta-optimization technique has been applied to tune selected parameters of RCQiEA algorithms and preliminary results of this approach have been presented.

*Meta-optimization*[4] is a method of tuning one optimization algorithm using another optimizer. Thus, tuning is considered as an optimization problem on its own. Meta-optimization is also known in the literature as meta-evolution, super-optimization, automated parameter calibration etc. First proposals of such approach have been presented in the late 1970s [32] for finding optimal parameter settings of a genetic algorithm. Other examples of genetic algorithm parameters meta-optimization include [14,24,31]. A state-of-the-art technique in this field is due to Pedersen[37]. The *Pedersen's tuning* is based on a simple optimization algorithm, Local Unimodal Sampling (LUS) [39]. This technique has been already used successfully to find good parameters for the evolutionary computing

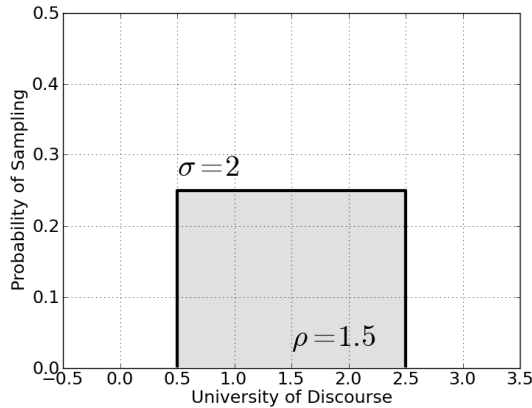
methods, Differential Evolution[36,37] and Particle Swarm Optimization[31,35,37], and the results turned out to generalize well to various optimization problems.

This paper is structured as follows. In section 2, main concepts of Quantum-Inspired Evolutionary Algorithm for problems coded in real numbers have been presented. In section 3, a Pedersen's meta-optimization technique has been briefly described. In section 4, conditions of conducted numerical experiments have been given, which allow the reader to replicate and to verify our research. In section 5, results of the experiments have been presented and evaluated. In section 6, final conclusions have been drawn and the article has been briefly summarized.

## 2. Quantum-Inspired Evolutionary Algorithm

This section briefly presents the Quantum-Inspired Evolutionary Algorithm for problem coded in real numbers (RCQiEA). For more detailed description of the algorithm the reader is referred to [5].

Let us denote by  $Q(t) = \{q_1, q_2, \dots, q_N\}$  the  $t$ -th generation of a *quantum population* consisting of  $N$  quantum individuals. Each *quantum individual*  $q_i$  is formed by  $G$  quantum genes  $g_i = [g_{i1} g_{i2} \dots g_{iG}]$ . A *quantum gene* in RCQiEA algorithm is modelled as a continuous, uniform probability distribution of sampling the university of discourse of the gene. Thus, the quantum gene can be encoded by two numbers  $g_{ij} = (\rho_{ij}, \sigma_{ij})$ , where  $\rho_{ij}$  denotes the centre of the  $j$ -th gene square pulse (mean value of probability distribution) and  $\sigma_{ij}$  denotes the width of the pulse. An illustration of a quantum gene has been depicted in figure 1.

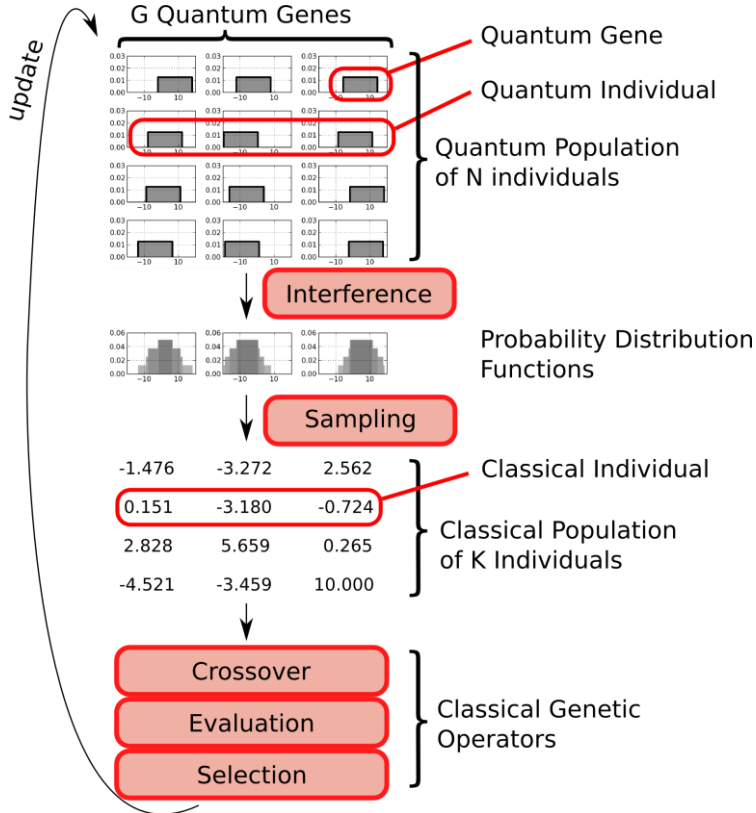


**Figure 1.** Example of real-coded quantum gene  $g_{ij} = (1.5, 2)$

The heights of square pulses are calculated according to the formula:

$$h_{ij} = \frac{1/\sigma_{ij}}{N} \quad (2.1)$$

which guarantees that later in the algorithm the total area created by the sum of the pulses on the same locus is 1. Consequently, the sum constitutes a probability distribution function (PDF) directly.



**Figure 2.** General overview of Real-Coded Quantum-Inspired Evolutionary Algorithm

The general schema of RCQiEA algorithm has been illustrated in figure 2. Briefly, the quantum population  $Q(t)$  encodes a probability distribution function for each variable. The algorithm samples the search-space with respect to these distributions. Then, typical recombination, evaluation and selection of the resulting classical individuals are performed. Finally, probability distributions encoded in  $Q(t+1)$  are updated to increase the chance of sampling good individuals in future generations. The full pseudocode of the algorithm is given as follows:

1.  $t \leftarrow 1$
2. create the initial quantum population  $Q(1)$  with  $N$  individuals, consisting of  $G$  quantum genes each // *initialization*
3. **while**  $t \leq T$ : // *main loop*
4.     create a cumulative distribution functions (CDF) for the population  $Q(t)$
5.     generate  $E(t)$  by observing  $Q(t)$  (using the CDFs)
6.     **if**  $t = 1$ :
7.          $C(t) \leftarrow E(t)$
8.     **else**:
9.          $E(t) \leftarrow$  crossover between  $E(t)$  and  $C(t)$  // *classical gen. operator*
10.         evaluate  $E(t)$  // *individuals' fitness evaluation*
11.          $C(t) \leftarrow K$  best individuals from  $[E(t) + E(t)]$  // *hard selection*
12.     **end if**
13.     **with** the  $N$  better individuals from  $C(t)$  **do** // *update of q. population*
14.          $Q(t+1) \leftarrow$  apply translate operation to  $Q(t)$
15.          $Q(t+1) \leftarrow$  apply resize operation to  $Q(t+1)$
16.     **end with**
17. **end while**

In the beginning of the algorithm (step 2), the initial population  $Q(1)$  is created consisting of individuals with random pulses centres. The width  $\sigma$  of all genes is set initially to the total domain width. In the beginning of the main loop (steps 4 and 5), the quantum population  $Q(t)$  is *observed*, i.e. a classical population  $E(t)$  of  $K \geq N$  individuals is created. This step is performed in two stages. First, *interference process* between the quantum individuals is simulated and  $G$  probability distribution functions are created by summing the individuals' pulses of the same loci. Then, corresponding cumulative distribution functions are calculated which allows sampling the search-space with respect to the probability distributions easily, as follows:

$$x = CDF^{-1}(r)$$

where  $r$  is a random number in the  $[0,1]$  interval,  $CDF^{-1}$  is the inverse for the cumulative distribution function and  $x \in \mathbb{R}$  is an observed value of the variable.

In the first iteration of the algorithm, the observed population  $E(t)$  is copied directly to the population  $C(t)$  (step 7). In consequent iterations, an additional genetic operator, classical uniform crossover between currently observed popula-

tion  $E(t)$  and the previously generated population  $C(t)$ , is applied. The pseudocode for the crossover operator is as follows:

1. **for**  $i = 1$  to  $K$ :
2.     select individual  $e_i$  from  $E(t)$
3.     select individual  $c_i$  from  $C(t)$
4.     **for**  $j = 1$  to  $G$ :
5.          $r \leftarrow$  random number in  $[0,1)$  using uniform distribution
6.         **if**  $r < \xi$  :
7.              $e'_{ij} \leftarrow e_{ij}$
8.         **else:**
9.              $e'_{ij} \leftarrow c_{ij}$
10.         **end if**
11.     **end for**
12. **end for**

The parameter  $\xi$  (in line 6 above) is the *crossover rate*. The value  $\xi = 1$  carries on all the genes in the created individuals to the offspring. The crossover rate equal to 0 does not modify the individuals and no evolution occurs.

The last step in the main loop of RCQiEA algorithm is an update of the quantum population  $Q(t)$ . This operation consists of two stages. First, the centre  $\rho_{ij}$  of each gene are homogenously modified by making the mean values of genes equal to the values of the genes from the  $N$  best individuals from the classical population  $C(t)$ . Then, the resize operation is applied, as given in equation:

$$\sigma_{ij} = \begin{cases} \sigma_{ij} \cdot \delta & \text{if } \varphi < 1/5 \\ \sigma_{ij} / \delta & \text{if } \varphi > 1/5 \\ \sigma_{ij} & \text{if } \varphi = 1/5 \end{cases} \quad (2.2)$$

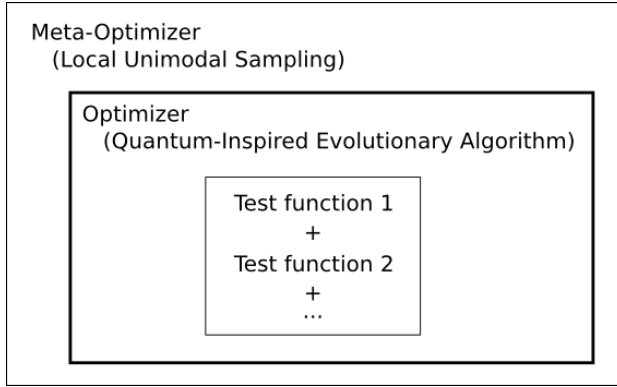
where:  $\varphi$  is the rate of how many individuals of the current  $C(t)$  population have their fitness improved in comparison to the previous generation and  $\delta \in (0,1)$  is the *contraction factor*. The formula (2.2) is based on a simple heuristic known as the 1/5th rule: If more than 20% of individuals improve their fitness, the range of sampling the search space is increased. If less than 20% of individuals improve their fitness, the range of sampling the search space is decreased. In other words, if only few individuals improve in subsequent generations, the population starts to converge and the algorithm performs local search. When the pulses widths are modified, also their height must be updated, according to the equation (2.1), because the sum of their areas needs to form a valid probability distribution function. The pulses resize operation can be performed at

every iteration or it can be applied at larger steps, specified by the parameter  $k$ . For example, if the parameter  $k = 5$ , the width of the pulses is modified every five generations of the population  $Q(t)$ .

### 3. Meta-optimization and Local Unimodal Sampling

Performance of evolutionary computing methods depends strongly on various behavioural parameters, such as mutation range, genetic operators application probabilities, learning factors, migration rate, selection pressure etc. Usually, the parameters have great impact on performance and efficacy of the algorithm and often they are found by manual, time-consuming experimentation. Parameters values can be evaluated according to the performance of the algorithm. Such *meta-fitness* of an algorithm is a quantity that describes performance of the algorithm with given parameters set. When the number of behavioural parameters increases, the time usage for computing such a *performance landscape* increases exponentially. Moreover, because of stochastic nature of evolutionary algorithms, the algorithm meta-fitness needs to be based on an average over at least 50 executions of the algorithm [27]. Consequently, evaluation of meta-fitness is a very time-consuming and computationally exhaustive operation. For example, if a population consists of 100 individuals evolving for 1000 generations, one evaluation of the evolutionary algorithm meta-fitness requires  $50 \cdot 100 \cdot 1000 = 5000000$  evaluations of the individuals' fitness. As it is easy to see, an evaluation of meta-fitness takes several orders of magnitude longer than an evaluation of fitness. Therefore, an efficient method is needed to search the space of parameters.

*Meta-optimization* is a systematic approach to this problem and the general idea behind it has been presented in figure 3. The meta-optimization algorithm assesses parameters of the underlying optimizer according to its meta-fitness measure. Assuming that the global optimum of the fitness function is known, the meta-fitness can be based on the number of iterations that the algorithm needs to find the correct solution and this method has been used in this paper.



**Figure 3.** Meta-optimization concept

A modern technique of meta-optimization is due to Pedersen[35,36,37]. Pedersen's tuning is based on Local Unimodal Sampling (LUS) algorithm which has been introduced in [38] and it was initially designed for unimodal functions, although it has been found to work well for harder optimization problems. Compared to the traversal of the entire parameters space, the meta-optimization technique uses only a small fraction of the computational time. It is critical for meta-optimization due to the computational time needed in the meta-optimization process. Pedersen's tuning technique is not limited only to his proposal of the overlaid meta-optimizer, Local Unimodal Sampling. The technique includes also, for example, a *pre-emptive fitness evaluation* which allows reducing computational time significantly if the meta-fitness is based on aggregate performance for several problems (simultaneously tuning parameters with regard to multiple optimization problems).

Let us denote by  $\vec{p}$  a vector of the algorithm parameters that are going to be tuned. The pseudocode of Local Unimodal Sampling is as follows:

1. Initialize  $\vec{p}$  to a random solution in the parameters search space:

$$\vec{p} \leftarrow U(\vec{b}_{lo}, \vec{b}_{up})$$

2. Set the initial sampling range:

$$\vec{d} \leftarrow \vec{b}_{up} - \vec{b}_{lo}$$

3. **Repeat** until the termination criterion is met:

- a.  $\vec{a} \leftarrow U(-\vec{d}, \vec{d})$

- b.  $\vec{y} = \vec{p} + \vec{a}$

- c. **if**  $f(\vec{y}) < f(\vec{x})$ , **then:**  $\vec{x} \leftarrow \vec{y}$

- d. **else:**  $\vec{d} \leftarrow q \cdot \vec{d}$ , where  $q = \sqrt[n]{1/2} = 2^{-\beta/n}$

where:  $n$  is the number of the tuned parameters,  $U$  denotes a random vector from  $n$ -dimensional hypercube,  $\bar{b}_l$  and  $\bar{b}_{up}$  are lower and upper boundaries of the parameters and  $\beta$  is a parameter of the Local Unimodal Sampling algorithm.

In contrast to most evolutionary computing methods, LUS is not a population-based algorithm. Only one candidate solution  $\bar{p}$  is iteratively evaluated and updated, which behaves very similarly to (1+1) evolution strategy. Consequently, LUS is often able to find the optimum in comparatively few evaluations of fitness function in search-spaces that are fairly smooth.

## 4. Numerical Experiments

In [5], it has been demonstrated that RCQiEA outperforms several modern metaheuristics, Stochastic Genetic Algorithm, Fast Evolutionary Programming, Fast Evolutionary Strategy and Particle Swarm Optimization, in numerical optimization. However, no general rules have been given on setting values of the algorithm parameters. Neither the authors of [5] provided the parameter  $\delta$  they have used in their experiments.

This section describes conditions of the experiments which replicate and extend previous studies. Objective of our experiments was to analyse influence of the parameters  $\bar{p} = (\xi, \delta)$  and to find their optimum values in the search space  $[0,1] \times [0,1]$ . The algorithms, RCQiEA and LUS, have been implemented in Python programming language. To accelerate the computations, objective functions have been implemented in C, a lower-level programming language. As a random number generator the Mersenne twister algorithm has been used which provides very good statistical properties and an ultra-long period which are essential for reliable experimentation with stochastic algorithms[27,30]. The experiments have been conducted on i686 machine with Intel(R) Core i7 CPU (2.93 GHz), 12 GB RAM.

Performance of the two algorithms has been compared: RCQiEA with the initial parameters given in table 2 and RCQiEA algorithm tuned with the meta-optimization technique. Because the value of  $\delta$  was not provided in [5],  $\delta = 0.5$  was assumed for the comparison. Four test functions have been used. The functions, their domains and dimensionality have been presented in table 1. The global minimums of the functions are located in  $x^* = (0, 0, \dots, 0)$ , where  $f(x^*) = 0$ . The experiments concerned optimization of 30-dimensional functions  $f_1 - f_4$ . However, to illustrate complexity and modality of the functions, their two-dimensional versions have been presented in figure 4.  $f_1$  and  $f_2$  are unimodal and  $f_3$  and  $f_4$  are multimodal functions. Parameter  $\beta$  of the overlaid LUS algorithm

was set to  $\frac{1}{3}$ , which has been drawn from [35]. 30 iterations of the algorithm have been executed.

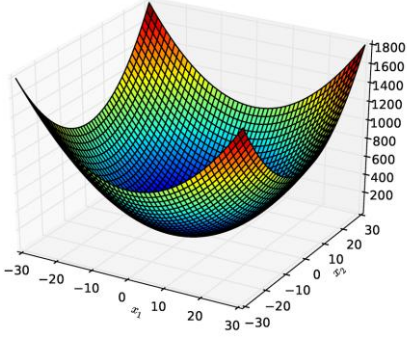
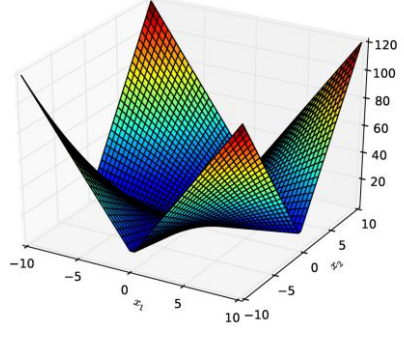
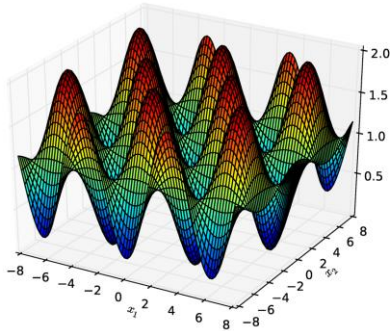
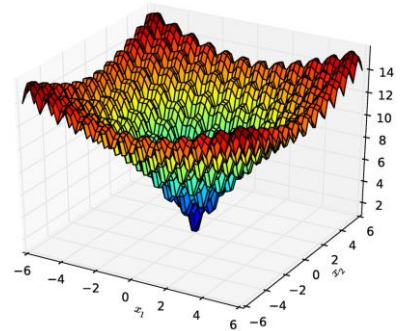
The meta-fitness  $\tilde{f}(\xi, \delta) : [0,1] \times (0,1] \mapsto \mathbb{R}^+$  has been based on the test functions  $f$  evaluation count. When RCQiEA algorithm finds the optimum solution  $x^*$  with a satisfactory precision  $\varepsilon = 10^{-6}$ , i.e.  $|f(x) - f(x^*)| < \varepsilon$ , further optimization of the test function is aborted and the evaluation count of the  $f$  function is taken into account. An average of this evaluation count is taken over 50 runs of RCQiEA algorithm as the value of the algorithm meta-fitness  $\tilde{f}$ . Moreover, if the algorithm failed to locate optimum solution  $x^*$  in 5000 iterations, the penalty of 100000 was added.

**Table 1.** Test functions used in the experiments

Equation	Domain	Dimension	$f_{\min}$
$f_1(x) = \sum_{j=1}^{30} x_j^2$	$x_j \in [-30, 30]$	30	0
$f_2(x) = \sum_{j=1}^{30}  x_j  + \prod_{j=1}^{30}  x_j $	$x_j \in [-10, 10]$	30	0
$f_3(x) = \frac{1}{4000} \sum_{j=1}^{30} x_j^2 - \prod_{j=1}^{30} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1$	$x_j \in [-600, 600]$	30	0
$f_4(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{30} \sum_{j=1}^{30} x_j^2}\right) - \exp\left(\frac{1}{30} \sum_{j=1}^{30} \cos(2\pi x_j)\right) + 20 + e$	$x_j \in [-32, 32]$	30	0

**Table 2.** Parameters of the RCQiEA algorithm drawn from [5]

	$f_1$	$f_2$	$f_3$	$f_4$
<b>Size (N) of the quantum population <math>Q(t)</math></b>	10	5	10	4
<b>Size (K) of the classical population <math>C(t)</math></b>	10	5	10	4
<b>Number of generations <math>k</math> before checking improvement</b>	10	8	5	20
<b>Crossover rate <math>\xi</math></b>	0.1	0.1	0.1	1
<b>Contraction factor <math>\delta</math></b>	0.5	0.5	0.5	0.5
<b>Maximum number of generations</b>	5000			

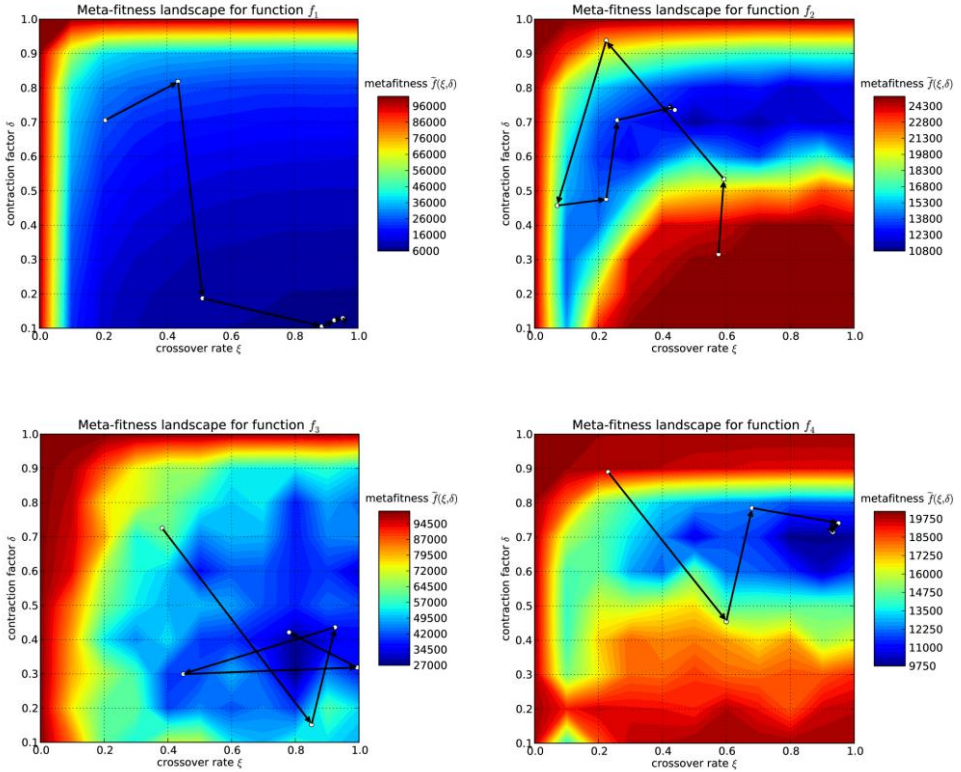
$f_1$  $f_2$  $f_3$  $f_4$ 

**Figure 4.** Two-dimensional version of the test functions  $f_1 - f_4$ .

## 5. Results

For the four test functions, meta-fitness  $\tilde{f}$  landscapes of the algorithm RCQiEA have been approximated and they have been depicted in figure 5. The plots of the landscapes have been approximated on a grid of points with the step of 0.1 in each dimension which results in quite low precision but provides a general overview of the landscape shape. The approximation of the landscapes took about one day on a modern personal computer. To accelerate the computation, the search-space  $[0,1] \times (0,1)$  has been divided into 8 regions. The computation for each region has been performed in parallel in separate thread which reduced the computational time greatly on the processor providing 4 cores and Hyper-Threading Technology. It is easy to see that the meta-fitness  $\tilde{f}$  landscapes for the

considered test functions are almost smooth and regular. In figure 5, selected trajectories of LUS algorithm in the meta-fitness search space have been also presented.



**Figure 5.** Meta-fitness landscapes for the four test functions

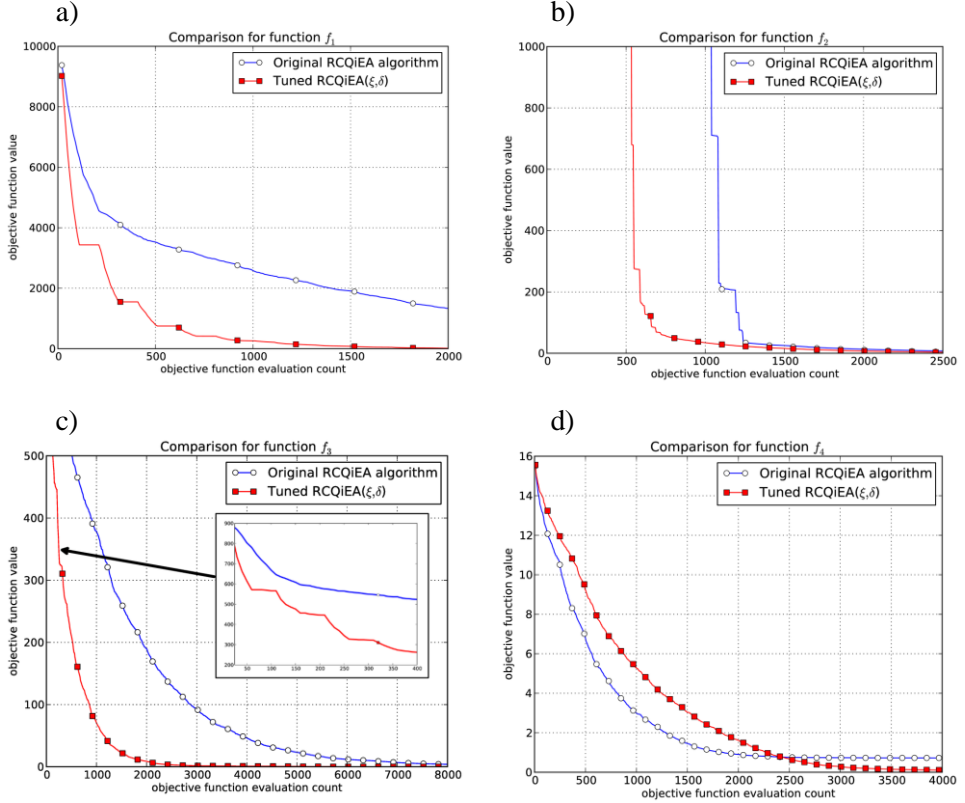
The best parameters  $\xi^*$ ,  $\delta^*$  that eventually have been found with LUS algorithm for the test functions have been presented in table 3.

**Table 3.** Best parameters values which have been found by meta-optimization process

	$f_1$	$f_2$	$f_3$	$f_4$
<b>crossover rate</b> $\xi^*$	0.952	0.439	0.780	0.933
<b>contraction factor</b> $\delta^*$	0.129	0.735	0.421	0.714

The results of the comparison for the functions  $f_1 - f_4$  have been presented in figure 6. For the functions  $f_1, f_2$  and  $f_3$ , the tuned algorithm outperformed the initial algorithm in both convergence rate and the quality of the best found solution. For the function  $f_4$ , the algorithm had a lower convergence rate in the beginning but eventually it finds a better solution of the optimization problem than the

original algorithm. It is due to the definition of the metafitness  $\tilde{f}$ , i.e. the goal of the meta-optimizer was to find the parameters set that makes the RCQIEA algorithm to find the optimum solution precisely.



**Figure 6.** Comparison of the algorithm efficacy with original and tuned parameters for the test functions: a)  $f_1$  b)  $f_2$  c)  $f_3$  d)  $f_4$

Despite the plots have been generated for an average over 50 runs of the algorithms, an unexpected pattern emerged. A suspicious periodicity of the evolution can be observed in subfigure 6a and it needs some explanation. In the beginning of the evolution, the tuned algorithm performs very well and most individuals improve. Thus, after the first check for the improvement (in  $k$ -th generation), the width of pulses is extended by a large factor (division by  $\delta = 0.129$ ), according to equation (2.2). Consequently, in the next period of  $k$  generations, the algorithm samples a much bigger space (namely,  $(1/0.129)^{30} \approx 4.8 \cdot 10^{26}$  times bigger) and it is unlikely that any individual improves. After the next  $k$  generations, the improvement rate is checked again. This time, the width of each pulse is multiplied by  $\delta = 0.129$  and the oscillation continues. The period length is  $k$  genera-

tions ( $k \cdot K$  objective function evaluations). The conclusion can be drawn that for small values of the contraction factor  $\delta$  and highly dimensional objective functions the algorithm modifies the sampling range back and forth what creates favourable conditions for the periodical behaviour of the evolution, i.e. periods of slow convergence alternate periods of fast convergence rate. The same behaviour can be also observed in subfigures 6b and 6c.

## 6. Conclusions

In this paper, a meta-optimization algorithm based on Local Unimodal Sampling has been applied to tune the two parameters of Quantum-Inspired Evolutionary Algorithm for problem coded in real numbers, crossover rate  $\xi$  and contraction factor  $\delta$ . Performance landscapes for the two parameters have been approximated. Contrary to traversal of the entire parameters search space or manual experimentation, Local Unimodal Sampling is an effective method in tuning quantum-inspired evolutionary algorithms. Possibilities for further research in this field include: other test functions, optimization of other parameters of the algorithm. Also, the experiments should be conducted with same values of other parameters, which would allow identifying general rules for setting crossover rate and contraction factor.

## 7. References

- [1] Alfares, F., Esat, I.: Real-coded quantum inspired evolution algorithm applied to engineering optimization problems, Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2006. pp. 169-176, 2008
- [2] Al-Othman, A., Al-Fares, F., EL-Nagger, K.: Power System Security Constrained Economic Dispatch Using Real Coded Quantum Inspired Evolution Algorithm, International Journal of Electrical, Computer, and Systems Engineering Vol. 1, 2007
- [3] Bin, L., Junan, Y., Zhenquan, Z.: GAQPR and its application in discovering frequent structures in time series, in Proceedings of the 2003 IEEE International Conference on Neural Networks & Signal Processing, Vol. 1, pages 399-403, 2003
- [4] Birattari, M., Tuning metaheuristics: a machine learning perspective, Springer Verlag, 2009
- [5] da Cruz, A., Vellasco, M., Pacheco, M.: Quantum-inspired evolutionary algorithm for numerical optimization, Quantum Inspired Intelligent Systems, pp. 115-132, Springer, 2008
- [6] da Cruz, A. V. A., Barbosa, C. R. H., Pacheco, M. A. C., Vellasco, M. B. R.: Quantum-inspired evolutionary algorithms and its application to numerical optimization problems., in ICONIP, edited by Pal, N. R., Kasabov, N., Mudi, R. K.,

- Pal, S., Parui, S. K., volume 3316 of Lecture Notes in Computer Science, pages 212-217, Springer, 2004
- [7] da Cruz, A. V. A., Pacheco, M. A. C., Vellasco, M. B. R., Barbosa, C. R. H.: Cultural operators for a quantum-inspired evolutionary algorithm applied to numerical optimization problems., in IWINAC (2), volume 3562 of Lecture Notes in Computer Science, pages 1-10, Springer, 2005
  - [8] Defoin-Platel, M., Schliebs, S., Kasabov, N.: A versatile quantum-inspired evolutionary algorithm, in Proceedings of the International Conference on Evolutionary Computation (Singapore, September 25-28, 2007). CEC, volume 7, 2007
  - [9] de Pinho, A., Vellasco, M., da Cruz, A., A new model for credit approval problems: A quantum-inspired neuro-evolutionary algorithm with binary-real representation, Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pages 445-450, 2009
  - [10] Fan, K., Brabazon, A., O'Sullivan, C., O'Neill, M.: Option pricing model calibration using a real-valued quantum-inspired evolutionary algorithm, in GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pages 1983-1990, New York, NY, USA, 2007, ACM
  - [11] Fan, K., Brabazon, A., O'Sullivan, C., O'Neill, M.: Quantum-Inspired Evolutionary Algorithms for Financial Data Analysis, in EvoWorkshops, pages 133-143, 2008
  - [12] Fan, K., O'Sullivan, C., Brabazon, A., O'Neill, M., McGarraghy, S.: Calibration of the VGSSD option pricing model using a quantum-inspired evolutionary algorithm, Quantum Inspired Intelligent Systems, Springer Verlag, 2008
  - [13] Fu, X., Ding, M., Zhou, C., Sun, Y.: Multi-threshold image segmentation with improved quantum-inspired genetic algorithm, in Proceedings of SPIE, volume 7495, 2009
  - [14] Grefenstette, J.: Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics, Vol. 16, pages 122-128, 1986
  - [15] Grover L.K.: A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, p. 212, 1996
  - [16] Gu, J., Cao, C., Jiao, B., Gu, X.: An improved quantum genetic algorithm for stochastic flexible scheduling problem with breakdown, in GEC '09: Proceedings of the first ACM IGEVO Summit on Genetic and Evolutionary Computation, pages 163-170, New York, NY, USA, 2009, ACM
  - [17] Han, K., H., Kim, J.H.: Analysis of Quantum-Inspired Evolutionary Algorithm, Proceedings of the 2001 International Conference on Artificial Intelligence, pp. 727-730, 2001
  - [18] Han, K., H., Kim, J.H.: On Setting The Parameters of Quantum-Inspired Evolutionary Algorithm for Practical Applications, Proceedings of IEEE Congress on Evolutionary Computing, pp. 178-184, 2003
  - [19] Han, K., H., Kim, J.H.: On The Analysis of The Quantum-Inspired Evolutionary Algorithm With a Single Individuau, IEEE Congress on Evolutionary Computation, pp. 16-21, 2006
  - [20] Han, K. Kim, J.: Genetic quantum algorithm and its application to combinatorial optimization problem, in Proceedings of the 2000 Congress on Evolutionary computation, volume 2, pages 1354-1360, Citeseer, 2000

- [21] Jang, J., Han, K., Kim, J.: Quantum-inspired evolutionary algorithm-based face verification, *Lecture Notes in Computer Science* (2003)
- [22] Jeong, Y., Park, J., Shin, J., Lee, K.: A Thermal Unit Commitment Approach Using an Improved Quantum Evolutionary Algorithm, *Electric Power Components and Systems*, Vol. 37, 2009
- [23] Jopek, Ł., Nowotniak, R., Postolski, M., L.>About., Janaszewski, M.: Zastosowanie kwantowych algorytmów genetycznych do selekcji cech, *Scient. Bulletin of Academy of Science and Technology, Automatics*, 2010
- [24] Keane, A.: Genetic algorithm optimization in multi-peak problems: studies in convergence and robustness, *Artificial Intelligence in Engineering*, Vol. 9, 1995
- [25] Khosraviani, M., Pour-Mozafari, S., Ebadzadeh, M.M.: Convergence analysis of quantum-inspired genetic algorithms with the population of a single individual, *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 1115-1116, 2008
- [26] Lin, D. Waller, S.: A quantum-inspired genetic algorithm for dynamic continuous network design problem, *Transportation Letters: The International Journal of Transportation Research*, Vol. 1, 2009
- [27] Luke, S.: *Essentials of Metaheuristics*, 2009, available at <http://cs.gmu.edu/~sean/book/metaheuristics/>. Access date: October 1, 2010
- [28] Mahdabi, P., Abadi, M., Jalili, S.: A novel quantum-inspired evolutionary algorithm for solving combinatorial optimization problems, in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1807-1808, ACM, 2009
- [29] Mani, A. Patvardhan, C.: Solving Ceramic Grinding Optimization Problem by Adaptive Quantum Evolutionary Algorithm, *Intelligent Systems, Modelling and Simulation, International Conference on*, 2010
- [30] Matsumoto, M. Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 1998
- [31] Meissner, M., Schmuker, M., Schneider, G.: Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training, *Bmc Bioinformatics*, Vol. 7, 2006
- [32] Mercer, R. Sampson, J.: Adaptive search using a reproductive meta-plan, Master's thesis, University of Alberta, 1977
- [33] Narayanan, A. Moore, M.: Quantum-inspired genetic algorithms, in *Evolutionary Computation*, 1996., *Proceedings of IEEE International Conference on*, pages 61-66, 1996
- [34] Nielsen, M., Chuang, I. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2000
- [35] Pedersen, M.E.H., Chipperfield, A.J.: Simplifying particle swarm optimization, *Applied Soft Computing*, Vol. 10, pp. 618-628, 2010
- [36] Pedersen, M.E.H.: Good parameters for differential evolution. Technical Report HL1002, Hvas Laboratories, 2010
- [37] Pedersen, M.E.H.: Tuning & Simplifying Heuristical Optimization, PhD thesis, University of Southampton, School of Engineering Sciences, 2010

- [38] Pedersen, M.E.H.: Local unimodal sampling. Technical Report HL0801, Hvass Laboratories, 2008
- [39] Qin, C., Liu, Y., Zheng, J.: A real-coded quantum-inspired evolutionary algorithm for global numerical optimization, in Cybernetics and Intelligent Systems, 2008 IEEE Conference on, pages 1160-1164, 2008
- [40] Shor, P. W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput. Vol. 26, pp. 1484-1509, 1997
- [41] Talbi, H., Batouche, M., Draa, A.: A Quantum-Inspired Evolutionary Algorithm for Multiobjective Image Segmentation, International Journal of Mathematical, Physical and Engineering Sciences, Vol 1, pages 109-114, 2007
- [42] Talbi, H., Batouche, M., Draa, A.: A quantum-inspired genetic algorithm for multi-source affine image registration, Lecture Notes in Computer Science, 147-154, 2004
- [43] Udrescu, M., Prodan, L., Vladutiu, M.: Implementing quantum genetic algorithms: a solution based on grover's algorithm, in CF '06: Proceedings of the 3rd conference on Computing frontiers, pages 71-82, New York, NY, USA, 2006, ACM
- [44] Vlachogiannis, J. Lee, K.: Quantum-Inspired Evolutionary Algorithm for Real and Reactive Power Dispatch, IEEE Transactions on Power Systems, Vol. 23, 2008
- [45] Xing, H., Ji, Y., Bai, L., Liu, X., Qu, Z., Wang, X.: An adaptive-evolution-based quantum-inspired evolutionary algorithm for QoS multicasting in IP/DWDM networks, Computer Communications, Vol. 32, 2009
- [46] Xing, H., Liu, X., Jin, X., Bai, L., Ji, Y.: A multi-granularity evolution based Quantum Genetic Algorithm for QoS multicast routing problem in WDM networks, Computer Communications, Vol. 32, 2009
- [47] Zhao, S., Xu, G., Tao, T., Liang, L.: Real-coded chaotic quantum-inspired genetic algorithm for training of fuzzy neural networks, Computers and Mathematics with Applications, Vol. 57, 2009
- [48] Zheng, T. Yamashiro, M.: Solving flow shop scheduling problems by quantum differential evolutionary algorithm, The International Journal of Advanced Manufacturing Technology, 2010