An Improved Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) for Scheduling of Real-Time Task in Multiprocessor...

Article *in* Applied Soft Computing · December 2016

DOI: 10.1016/j.asoc.2016.12.051

CITATIONS
0

READS
53

5 authors, including:

Debanjan Konar
Sikkim Manipal Institute of Technology
18 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Siddhartha Bhattacharyya
RCC Institute of Technology
181 PUBLICATIONS **460** CITATIONS
SEE PROFILE

Kalpana Sharma
Sikkim Manipal University
39 PUBLICATIONS **164** CITATIONS
SEE PROFILE

Sri Raj Pradhan
4 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Applied Soft Computing 53 (2017) 296–307

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

An improved Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) for scheduling of real-time task in multiprocessor system

Debanjan Konar[a], Siddhartha Bhattacharyya[b,*], Kalpana Sharma[a], Sital Sharma[c], Sri Raj Pradhan[a]

[a] Department of Computer Science & Engineering, Sikkim Manipal Institute of Technology, Majitar, East Sikkim, Sikkim 7373136, India
[b] Department of Information Technology, RCC Institute of Information Technology, Canal South Road, Beliaghata, Kolkata 700015, India
[c] Department of Computer Science & Engineering, Manipal Institute of Technology, Manipal, Karnataka 576104, India

ARTICLE INFO

Article history:
Received 3 April 2016
Received in revised form 9 November 2016
Accepted 28 December 2016
Available online 4 January 2017

Keywords:
Multiprocessor system
Real-time task scheduling
EDF
SCTF
CGA
NP complete problem
QIGA
HPSO
Hybrid Quantum-Inspired Genetic Algorithm

ABSTRACT

This article concerns an efficient real-time task scheduling assisted by Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) in multiprocessor environment. Relying on concepts and the principles of quantum mechanics, HQIGA explores the computing power of quantum computation. To drive schedule toward better convergence, HQIGA operates using rotation gate for exploration of variable chromosomes described by *qubits* in Hilbert hyperspace. A fitness function associated with popularly known heuristic earliest deadline first (EDF) is employed and random key distribution is adopted to convert the *qubits* chromosomes to valid schedule solutions. In addition to this, permutation based trimming technique is applied to diversify the population which yields good quality schedules. To establish the effectiveness of the suggested HQIGA, it demonstrates using various number of real-time tasks and processors along with arbitrary processing time. Simulation result shows that HQIGA outperforms the classical genetic algorithm (CGA) and Hybrid Particle Swarm Optimization (HPSO) in terms of fitness values obtained using less number of generations and also it improves the scheduling time significantly. HQIGA is also tested separately with the heuristic Shortest Computation Time First (SCTF) technique to show the superiority of EDF over SCTF.

1. Introduction

Real-time tasks scheduling in multiprocessor environment always remains as daunting task for researchers in the field of optimization engineering. Multiprocessor systems have been emerged as powerful and popular computing means for processing real-time applications. Real-time tasks have been employed in some prominent areas of application such as process control system, manufacturing automation, embedded systems, telecommunication systems, robotics, multimedia applications. In the last decades, the applications on real-time tasks gradually tend to sophisticated and behaviorally complex and time constrained. On the contrary, the main challenges evolved by soft real-time applications in multiprocessor environment have been addressed to tackle those problems. However, in the context of real-time task scheduling, the research on real-time systems has been rejuvenated in multiprocessor environments. Owing to the inherent parallelism offered by the multi-processor environment, scheduling of real-time tasks is essentially efficient enough. Scheduling real-time tasks on a multiprocessor system can be characterized as sequencing the tasks in proper order and allows them to execute on the processors in which the task is assigned such that each task said to be meeting deadline constraint. Since last decade, a plethora of approaches have been aimed to schedule real-time tasks in multiprocessor environment. Examples include heuristic approaches [1–3], evolutionary algorithms [3–5] and hybrid techniques [6–9]. However, heuristic based techniques always do not produce optimal results in all circumstances. Various evolutionary based techniques [4,5,3,10–15] have been often applied to obtain valid scheduling of the tasks which is eventually a NP complete problem [16]. The average fitness value of a population can be improved by exhibiting Genetic algorithm and therein produces new population. The selection of suitable population size, fitness function, crossover and mutation probability are key factors behind the improvement in the average fitness value. Though classical genetic algorithm (CGA) based approaches stand in good stead in some cases, yet these approaches seldom pay any heed for satisfying deadline.

Amalgam of two classical domains, bio-inspired computing and quantum computing in the filed of computer science resulted a hybrid computing paradigm named quantum-inspired genetic

* Corresponding author.
E-mail address: dr.siddhartha.bhattacharyya@gmail.com (S. Bhattacharyya).

http://dx.doi.org/10.1016/j.asoc.2016.12.051
1568-4946/

algorithms (QIGA). Quantum computing is an emerging field of engineering depending on quantum mechanical effects to solve computational problems [17]. Using these such unique quantum mechanical phenomena, various optimization problems can be solved efficiently. On the contrary to some novel quantum algorithms [18–21] considered, these algorithms can be simulated on classical computers with great precision. However, it is extremely computationally exhaustive. Nevertheless, inherent randomness offered by representation of *qubits* drawn from the basic properties and concepts of quantum mechanics. First evolutionary algorithm inspired by quantum computing was proposed by A. Narayan et al. [22] in 1996 and this field is still vigorously studied now-a-days. For efficient implementation of these algorithms, quantum environment is not necessary, however, real demonstration of the algorithms on a quantum computer remain to investigate in near future and many researchers are working in this direction.

The remaining portion of the article is organized as follows. A comprehensive survey of the works associated with scheduling of real-time tasks is provided in Section 2. Section 3 elucidates the focus of research work of this article. Section 4 explains the foundations of real-time system, real-time task model and necessary objective function relevant to this article. Section 5 reflects the problem definition. The basic fundamentals of quantum computing necessary for understanding of quantum-inspired genetic algorithm (QIGA) are illustrated in Section 6. The principle of operation of quantum-inspired genetic algorithm (QIGA) [24–27] is presented in Section 7. Section 8 throws light into the operation of suggested hybrid quantum inspired genetic algorithm (HQIGA). The simulation results of application of the proposed HQIGA, CGA [28] and HPSO [29] with uniform priority on various numbers of randomly generated real-time tasks and these are assigned to arbitrary number of processors (variable size chromosome) illustrated in Section 9. Section 10 concludes the article with future directions in research.

2. Review of literature

The most common method adopted to tackle the multiprocessor scheduling is list scheduling since the beginning of the research on this field. Notable contributions in this direction are by Kwok et al. [30]. In this paper, an in-depth survey and classification of deterministic scheduling algorithms are presented. List scheduling approaches have been enormously adopted in task scheduling algorithms [31]. There are also few well-established scheduling heuristics suitable for task list scheduling to optimize inter-process communication delays. Examples include Insertion Scheduling Heuristic (ISH) and Duplication Scheduling Heuristic (DSH) [32,33].

A genetic algorithm assisted scheduling algorithm proposed by E.S.H. Hou et al. [14], tackles scheduling of tasks for a directed task graph in multi-processor environment. This algorithm yielded better results when compared with the other and governed by task graph with dependency but without communication delays. To obtain a feasible optimal solution, Annie S. Wu et al. [4] also presented a genetic algorithm relying on incremental fitness function. This procedure gradually improves the fitness values until a satisfactory solution is evolved. This approach is not applicable for large scale of tasks due to evaluating illegal schedules that may never become valid schedules. However, in spite of wide acceptance of classical genetic algorithm in the field of scheduling, it suffers from generating illegal schedules, therein leading to computational time overhead. To overcome this problem, R.C. Correa et al. [3] suggested a hybrid genetic algorithm introducing heuristics in the basic genetic operators in a classical genetic algorithm. In this approach, the authors presented a task duplication algorithm for reducing the scheduling time and they significantly improve the schedule by minimizing the number of invalid schedules. However, the inherent drawback with this method lies in the fact that this heuristic based approach under perform its classical counterpart in terms of computational time. Of late, M. R. Bonyadi et al.'s [5] approach a novel bipartite algorithm relying on genetic operations attained 10% less number of generations while compared with other techniques. The suggested GA inspired bipartite algorithm works on the characterization of maximum finish time of tasks in multiprocessor environment. A. Mahmood and A. Awadalla [34] presented a hybrid genetic algorithm composed with Modified List Scheduling Heuristic in real-time environment to produce feasible schedules incorporating well known heuristics Shortest Computation Time First (SCTF) and Earliest Dead-line First (EDF). Recently, S. C. Cheng et al. [12] achieved an optimal solutions to real-time tasks scheduling problem through a suitable feasible energy function incorporated in genetic algorithm. A genetic algorithm in parallel mode proposed by Moore [35] for scheduling of tasks in a distributed environment by comparing the results with mathematically predicted values, deserves special mention.

Currently, classical genetic algorithms (GA) have widely been adopted as a useful vehicle to tackle real-time tasks scheduling problems. Among the notable heterogeneous tasks scheduling techniques on heterogeneous processors, Page et al.'s [36] proposal, appropriate for scheduling in distributed environment. In spite of the wide acceptance of the classical genetic algorithm inspired approaches, these techniques fall short in generalization for producing valid schedules owing to the genetic operations (*crossover* and *mutation*) associated with them. Moreover, in real time scenario, classical genetic algorithm inspired algorithms often fail due to inherent time-consuming genetic operators incorporated in this algorithms.

Recent years witnessed the work proposed by Chen et al. [8] incorporating Ant Colony Optimization (ACO) to obtain feasible solutions and shown the superiority of the proposed ACO over GA in terms of processing time and percentage of successful scheduled solutions. However, ACO incurs high energy consumption compared to other evolutionary approaches. The Particle Swarm Optimization [37] is employed to solve the energy consumption problem by minimizing average utilization of the processor. It is found to be efficient than GA and ACO in terms of energy consumption. The problem with the PSO lies in the fact that it suits for some real-time task problems but it fails to meet the same kind of results in task assignment problem.

Enormous number of applications in quantum-inspired genetic algorithms (QIGA) are witnessed over the time in various domains, including image processing [38], flow shop scheduling [39–41], network design problems [42,43], thermal unit commitment [44], power system optimization [45]. Consequently, due to enormity in computational problems, it has been reported that QIGAs are superior to classical counterparts.

The first Quantum mechanical computers are invented in the year of 1980s [46] relying on the concepts of quantum computation and quantum algorithms. Many active efforts have been paid on quantum computers since on wide variety of computational domains it outperforms classical computing. There exists well-known quantum algorithms viz. Shors's quantum factoring algorithm [18,19] and Grover's quantum database search algorithm [20,21]. Owing to the inherent diversity offered by the quantum computation, the hybridization of quantum computation with evolutionary computing like classical genetic algorithms (CGA) is exploited. In order to process computational tasks exponentially faster, Quantum computing uses micro-quantum level effects which rely on the principles of quantum mechanics [47]. Quantum Inspired Genetic Algorithm (QIGA) [24–26,22], located in the intersection of quantum and evolutionary computation, is a emerging

among quantum soft computing researchers. Recent years have witnessed the work done by K.H. Han et al. [24–26],introduced QIGA for applying on optimization problems like flow-shop scheduling and Knapsack. A novel hybrid QIGA is also suggested by Bin-Bin Li et al. [27] aiming for permutation flow shop scheduling known as NP hard optimization problem. Due to the diversity in multiprocessor real-time scheduling, pure QIGA cannot be applied directly.

2.1. Motivation

To resolve wide variety of optimization problems like Knapsack problem, traveling sales person problem, scheduling real-time tasks in multiprocessor system, classical genetic algorithms (CGA) are routinely applied since it operates on a population (solutions) assisted by the survival of the fittest to generate subsequently better solutions in the context of approximations. However, due to inherent probabilistic representation, quantum inspired genetic algorithms (QIGA) are not paid enough attention among the quantum soft computing research communities. Researchers [24–27] have outlined basic quantum inspired-genetic algorithms, each novel approach contributes significantly on various optimization problem techniques, implementation and underlying model representation. In spite wide acceptance and well-establishment of quantum inspired genetic algorithms, these remain a daunting task as both theoretical and simulation tool sets are still in the inception stage of development and lack of maturity.

A common underlying thread shared by quantum inspired genetic algorithms till today is that each implements quantum bits or *qubits* and superposition of states (binary chromosomes). A novel hybrid quantum inspired genetic algorithm (HQIGA) is proposed in this current work by embedding the concepts of quantum computing. However, HQIGA uses a *qubit* due to probabilistic representation and obviates binary representation. The primary objective for representing the binary chromosome as *qubit* lies in the fact that linear superposition offered by QIGA drives the individual toward better solutions.

3. Proposed work

Assuming all real-time tasks are having uniform priority in execution, the current work suitably focuses to evolve an time efficient task scheduling in multiprocessor sense, referred as hybrid quantum inspired genetic algorithm (HQIGA). The concept of quantum-inspired genetic algorithm along with earliest deadline first (EDF) and Shortest Computation First (SCTF) are exhibited in HQIGA. In HQIGA, constituent bits of chromosomes with varying length are characterized as probabilistic representation *qubits*. The initial population of different real-time tasks to be scheduled are designated as quantum chromosome. Owing to inherent probabilistic nature of *qubits*, quantum chromosomes have better population diversity in schedules. The proposed work introduced quantum rotation gate to obviate crossover and mutation for updation of quantum chromosomes and to produces new population. The selection of suitable rotation angle is key behind the fast convergence of the schedules toward optimal solution. Quantum Rotation gate operation on the individual qubits of the chromosome exhibits new sequence of scheduled tasks and updating operation continues until all tasks are scheduled successfully. The task and processor sequencing is done by performing permutation of tasks and processors respectively. In this proposed work, EDF and SCTF are used to govern the fitness value for each chromosome individual. A quantum measurement is applied on each individual schedule, obtained using rotation gate operation.

In this proposed work, simulation result of HQIGA algorithm is presented using various number of real-time tasks application. The result obtained from classical genetic algorithm (CGA) and Hybrid Particle Swarm optimization (HPSO) has been compared with the proposed technique. A little investigation on comparative results reveals that the proposed HQIGA is superior to its classical counter part and HPSO in terms of fitness value and time.

4. Fundamentals of real-time systems

Real time system is characterized on the correct logical outcomes of computation given deadline constraint. Real time system relies on the correctness of results produced by the system and also on quantitative notion of time [48]. Guaranteed timing behavior offers predictability of the real-time tasks in terms of its completion time with certainty. In scheduling, the compromise of real-time tasks on quality of schedules is due to the deadlines associated with these tasks [2]. Real-time tasks can be classified soft real-time and hard real-time tasks depending on the consequence of task fails to satisfy their deadline constraint. Missing some deadlines are tolerable in case of soft real-time tasks. To address the problems faced by soft and hard real-time tasks, suitable scheduling algorithm is desirable. Moreover, the deadline of the tasks in real-time, it is also desirable for attainment of high utilization during the system meeting its deadlines [48].

4.1. Real-time task model

A real-time application basically comprises multiple task with various levels of criticality. In spite of the fact that it is desirable for real-time tasks to satisfy their deadlines, soft real-time tasks can be relived in some extent in context of missing deadlines without interrupting the system. However, it is obvious to paying penalty for some soft real-time tasks for not satisfying deadline constraint. Although, missing deadlines in hard real time systems are not tolerable at all; in turn, results may lead to undesirable or invalid (schedules). Firm real-time tasks, another class of real-time tasks, gain more rewards as the tasks finish their computation as early as possible [48]. Formally a real-time task model can be presented as follows.

A real-time system in multi-processor environment comprises with m number of real time tasks $T = \{T_1, T_2, T_3, \ldots, T_m\}$ and n identical processors $P = \{P_1, P_2, P_3, \ldots, P_n\}$. Each task $T_i \in T$ is characterized [34] by A_i: arrival time of the task T_i; R_i: ready time of the task T_i; C_i: worst case computation time of the task T_i; D_i: deadline of the task T_i; S_i: start time of the task T_i; F_i: completion time of the task T_i. The task T_i is said to meet its deadlines if $R_i \leqslant S_i \leqslant D_i - C_i$ and $R_i + C_i \leqslant S_i \leqslant D_i$ [15,49].

4.2. Objective function

In this article, given the optimization criteria earlier deadline first (EDF) and shortest computation first (SCTF), objective function evaluates the fitness value of each individual schedule. The shortest deadline first task and shortest computation first tasks is selected in every iteration of the scheduling in order to determine optimal schedule (chromosome) through EDF and SCTF respectively. Earliest deadline first (EDF) scheduling criteria [48] can defined for a set of real time tasks $T = \{T_1, T_2, T_3, \ldots, T_m\}$ as:

$$u_i = \sum_{j \in ST_i}^{n} u_{tj} \leqslant 1 \tag{1}$$

where the set of all tasks assigned to processor P_i is ST_i and u_{tj} is the utilization of processor P_i due to the task T_j. In this work, depending on the number of real-time tasks satisfying their deadlines, the

suggested fitness function or objective function determine fitness value for each schedule as $R_i \leqslant S_i \leqslant D_i - C_i$ and $R_i + C_i \leqslant S_i \leqslant D_i$.

5. Real-time task scheduling problem in multiprocessor environment

In this current work, an attempt has been made to solve the real time task scheduling problem in multiprocessor perspective. It is also aimed to allow maximum number of tasks for scheduling subjected to each task meeting its deadline. Formally, aforementioned real-time task scheduling problem can be characterized with m number of tasks in real-time environment and n number of identical processors as follows.

Each real time task of m tasks $T_i (1 \leqslant i \leqslant m)$ processed on processors $P_j (1 \leqslant j \leqslant n)$ assuming maximum one task can be processed on a processor. p_{ij} is the computation time of ith task on jth processor. There are plenty of tasks which are interrelated to other tasks, adopt twofold global scheduling strategies.

Firstly, in order to obviate inter-communication cost, dependent tasks are scheduled on the same processor for execution by employing the real time tasks with earlier dead line first order.

Secondly, the tasks which are arrived for complete execution, accumulated in ready queue and these tasks can be accessed by any processor therein.

5.1. Assumptions

To formulate real-time task scheduling, the following different assumptions have been made in multiprocessing environment. The problem is designed on real time task model [15,49] where arrival time, ready time, maximum computation time, deadline of all tasks are known in advance. Soft real time tasks are non-preemptive [50]. A task running on a particular processor cannot be preempted before it completes its execution. Each task is having equal priority and in every instance the proposed scheduler selects task from the task queue for execution. Each processor is associated with its own dispatch task queue.

6. Basics of quantum computing

Relying on the postulates of quantum mechanics, the field of research on quantum computing centered on computational devices. Various quantum mechanical operators viz. coherence, superposition, entanglement on the constituent basic states are characterized using the principles of quantum mechanics. In quantum mechanical system, several basic states or solutions often are combined in superposition form. Coherence refers to the correlation of the basic states. The property of entanglement gives rise to non-local interaction among bipartite correlated states.

6.1. Concept of qubits

The basic unit of information processing in quantum computing is a *qubit* or quantum bit [47]. Unlike classical bits (0 and 1), in *qubits* *eigenstates* $|0\rangle$ and $|1\rangle$ are superimposed. It can be represented as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

$|\alpha|^2$ and $|\beta|^2$ are known as probability for occurrence of $|0\rangle$ and $|1\rangle$ *eigenstates* respectively in the *qubit* $|\phi\rangle$ subjected to normalization criteria

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3}$$

where α and β are complex numbers. Linear unitary operators exploit the fundamental operations on *qubits* in Hilbert space. The essence of quantum computation for realizing several quantum logic gates [17] are implemented using these unitary operations.

6.2. Concept of rotation gate

A rotation gate with a rotation angle ω can be realized as

$$R(\omega) = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \tag{4}$$

To transform a *qubit*, this rotation gate operates as

$$\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \times \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{5}$$

The upgraded quantum bit (α', β') is obtained by employing a rotation angle ω on *qubit* (α, β).

6.3. Quantum measurement

A postulate [51] in quantum mechanics (QM) suggests that "if a coherent or linear superposition of basic states interacts with its environment, then on quantum measurement the superposition is destroyed". A wave function ψ, exists in the containment of set of states $|\phi_i\rangle$ in Hilbert hyperspace 0–1 describes a quantum systems as

$$|\psi\rangle = \sum_{j}^{n} p_j |\phi_j\rangle \tag{6}$$

Coherent of the basic states $|\phi_j\rangle$ is known as $|\psi\rangle$. p_j is complex coefficient and $|p_j|^2$ are the probability of ψ obtained using quantum measurement of $|\psi\rangle$. The wave function ψ corresponds a real physical system that must collapse [17] to exactly one basic state, therein the probabilities amplitudes $|p_j|^2$ sum to unity.

$$\sum_{j}^{n} |p_j|^2 = 1 \tag{7}$$

Superposition of basic states always form a coherent which deals with the physical system (by being measured) guided by the wave function ψ.

Using Dirac notation, the probability for a quantum state $|\psi\rangle$ being measured into an *eigenstate* $|\phi_i\rangle$, can be depicted as $|\langle\phi|\psi\rangle|^2$. For an example, consider a quantum state which is superpose of $|0\rangle$ and $|1\rangle$ as

$$|\phi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle \tag{8}$$

The coherent quantum state $|\phi\rangle$ is neither in state $|0\rangle$ nor in $|1\rangle$ where as in classical state it must be in one of the basic states. Therefore, using Born rule the probability of occurrence of $|1\rangle$ is in $|\phi\rangle$ as

$$|\langle 1|\psi\rangle|^2 = \frac{1}{4} \tag{9}$$

Similarly, the probability of occurrence of $|1\rangle$ in $|\phi\rangle$ is 3/4.

7. Quantum-Inspired Genetic Algorithm (QIGA)

The postulates of quantum mechanics inspire the idea of quantum inspired genetic algorithm to perform faster by exhibiting the inbuilt properties of quantum computation. The notable contribution in this field is by Han et al. [24–26]. They adopted *qubits* (Q bit) for better characteristics of individual chromosomes in QIGA instead of bits representation of chromosomes, outperforming in diversity of population than classical counterpart. In quantum chromosomes, linear superposition of all possible binary states provides

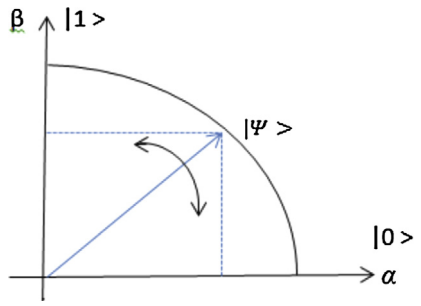


Fig. 1. Geometric demonstration of single bit quantum gene state (imaginary part is not shown for clarity).

the great advantage over classical representation. To converge the chromosome individuals toward optimal solutions in QIGA, quantum rotation gate is incorporated. The necessary basic organization of quantum chromosomes is presented in nutshell in the following section.

7.1. Quantum chromosome representation

In order to exploit randomness offered by the probabilistic models of quantum chromosomes (chromosomes described by *qubits*), constituent quantum genes are realized using *qubits* in quantum inspired genetic algorithm. A basic quantum gene [24] state $|\psi\rangle$ can be described as unit vector as shown in Fig. 1. The probability for finding values 0 and 1 are directly proportional to the angle between the vector and the horizontal axis.

The elements of each individual quantum chromosome C is presented as *qubits* string of size n.

$$C = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_n \\ \beta_1 & \beta_2 & \beta_3 & \dots & \beta_n \end{bmatrix} \tag{10}$$

where

$$|\alpha_i|^2 + |\beta_i|^2 = 1, i = 1, 2, 3, \dots, n. \tag{11}$$

The structure of quantum chromosomes containing quantum genes is demonstrated in Fig. 2. The main advantage of *qubit* representation over binary representation lies in linear superposition of quantum states. The following example include a trinity of qubits with trinity of pairs of amplitudes represented as

$$C_0 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{bmatrix} \tag{12}$$

Therefore, only one qubit individual is sufficient to characterize eight distinct binary states 000, 001, 010, 011, 100, 101, 110, 111.

The states of the system can be designated using tensor product as

$$\frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle - \frac{1}{4}|010\rangle - \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle - \frac{1}{4}|110\rangle - \frac{\sqrt{3}}{4}|111\rangle \tag{13}$$

The above results show probabilities of the states $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, and $|111\rangle$ are 1/16, 3/16, 1/16, 3/16, 1/16, 3/16, 1/16 and 3/16 respectively. In this aspect, *qubits* strings of length three comprises the information of eight states as tensor product. The inherent structure of quantum chromosome offers the key features in QIGA as shown in Fig. 2. Quantum-inspired genetic algorithms (QIGA) relies on the principle of evolutionary computing. This bio-inspired algorithm of QIGA demonstrated below [23].

Algorithm. Quantum Inspired Genetic Algorithm (QIGA)

1 Initialize the generation number as $i = 0$.
2 Initial population $Pop(i) = \{Q_1(i), Q_2(i), Q_3(i), Q_4(i), \dots Q_s(i)\}$ is produced randomly, where $Q_j(i)$ denotes the jth individual chromosome in the ith generation as follows

$$Q_j(i) = \begin{bmatrix} \alpha_1(i) & \alpha_2(i) & \alpha_3(i) & \dots & \alpha_n(i) \\ \beta_1(i) & \beta_{2(i)} & \beta_3(i) & \dots & \beta_n(i) \end{bmatrix}$$

3 Transform $Pop(i)$ in to corresponding solution $Sol(i)$ by exhibiting quantum measurement.
4 The best solution among $Sol(i)$ is accumulated into the storage $Best(i)$.
5 If termination condition is satisfied then save the best solution in $Best(i)$ else
6 do
7 $i = i + 1$
8 To update $Pop(i-1)$, apply rotation gate $R(\omega)$ and $Pop(i)$ is found to be as

$$Pop(i) = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} Pop(i-1)$$

9 Repeat the step no. 4.
10 Accumulation of the best solution among $Sol(i)$ and $Best(i-1)$ at $Best(i)$.
11 Until termination condition is false.

In the initial phase of QIGA, the quantum genes of all individuals chromosomes in the initial population are set to linear superposition of basic states with equal probability $1/\sqrt{2}$ as $(1/\sqrt{2}|0\rangle + 1/\sqrt{2}|1\rangle)$. Owing to *qubits* representation, fitness value evaluation of quantum chromosome individual is carried out by converting *qubits* into classical binary bits. Therefore, an algorithm of quantum measurement is required to convert the *qubits* into binary representation. The necessary genetic operations are performed on the individuals guided by quantum gate operator. Selection of suitable rotation angles leads to the efficiency of this algorithm.

8. Proposed Hybrid Quantum-Inspired Genetic Algorithm (HQIGA)

Since, Hybrid Quantum Inspired Genetic Algorithm is a replica of Quantum Inspired Genetic Algorithm, HQIGA follows same procedure as QIGA with added heuristic earlier deadline first (EDF) and some necessary submodules for scheduling. This suggested work is

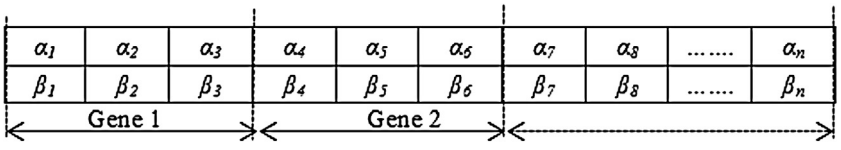


Fig. 2. Structure of quantum chromosome.

aimed to process each of m tasks in real time environment on n identical processors. Each task $T_i(1 \leqslant i \leqslant m)$ needs to be executed on the processors $P_j(1 \leqslant j \leqslant n)$ in interleaving manner. The processing time of task i on the processor j is p_{ij}. Select valid schedules which are meeting their deadline as $R_i \leqslant S_i \leqslant D_i - C_i$ and $R_i + C_i \leqslant S_i \leqslant D_i$ subjected to EDF schedule criteria

$$\sum_{j \in ST_i}^{n} u_{tj} \leqslant 1$$

The full procedure of the proposed HQIGA can be presented as follows

1 Start
2 Generate task queue for m real-time tasks. Assign arrival time A_i, ready time R_i, deadline D_i and computation time C_i for each real-time task i.
3 Do
4 Create a ready queue for the tasks arrived for execution.
5 If (ready queue size $\geqslant 1$)
6 Chromosome size (CS) is set to be the ready queue size.
7 $i \leftarrow 0$
8 Initial population of size s as $P_H(i) = \{P_1^i, P_2^i, \ldots, P_s^i\}$ is generated randomly in terms of quantum chromosomal where P_j^i represents the jth individual chromosome (string of qubits) in the ith generation
9 Each gene of quantum chromosome (valid population) is allocated to appropriate processor by quantum measurement using **Qmeasure**(x) followed by permutation based trimming operation
10 Initialize *Gen, Maxgen*
11 Quantum chromosome represents a valid population $P_H(i)$ and computed using the proposed objective function **objectivefunct**()
12 The best chromosome C among the population is selected and store it in $B_H(i)$
13 Do
14 To obtain new population $P_H(i+1)$, rotation gates are operated on *qubits* of the quantum population $P_H(i)$ with the best chromosome C.
15 Evaluate the new valid population $P_H(i+1)$ using the proposed objective function **objectivefunct**()
16 Choose the best chromosome among the population $P_H(i+1)$ and $B_H(i)$ and store it in $B_H(i+1)$
17 $i = i+1$ and go to step 13
18 Until the scheduling criteria is not satisfied for the best chromosome C and number of generation, i ! = *Maxgen*).
19 The best chromosome is scheduled in the corresponding processor.
20 Until size of the execution queue ! = *Maximum Size*(n).
21 Stop

Like QIGA, HQIGA is also a probabilistic algorithm relying on *qubit* individual of length n. Initially, randomly generated population of quantum chromosomes $P(i) = \{Q_1^i, Q_2^i, \ldots, Q_s^i\}$ at generation i, where Q_j^i represents the ith individual comprising of qubits defined as

$$Q_i(i) = \begin{bmatrix} \alpha_{j1}^i & \alpha_{j2}^i & \alpha_{j3}^i & \ldots & \alpha_{jn}^i \\ \beta_{j1}^i & \beta_{j2}^i & \beta_{j3}^i & \ldots & \beta_{jn}^i \end{bmatrix} \quad (14)$$

where n is number of processors as *qubits* in the system. Initially, Q_j^0 comprises equal probability $\frac{1}{\sqrt{2}}$ for all *qubits* and also jth *qubit* ($j = 1, 2, 3, \ldots, n$) of quantum chromosome [24] converted to a binary bit using probabilistic quantum measurement. Given a randomly generated number r within the range [0, 1); if $r < |\alpha_j|^2$, then corresponding binary bit of the string is said to be 1. In this way, a binary string of length n is formed as

Procedure Qmeasure(x)
1 Begin
2 $j = 0$
3 While ($j < n$) do
4 $j = j + 1$
5 If rnd[0, 1) $< |\alpha_j|^2$
5 Return 0
5 Else
6 Return 1
7 End if
8 End while
9 End

In this proposed HQIGA algorithm, the binary chromosome individual obtained using quantum measurement, follows random key representation [52,53]. Consider the system consists five processor (quantum gene) and trinity of bits in a string of *qubits* to represent a real-time task. Binary string for corresponding *qubit* string is [|100|101|110|111|110] and corresponding random key form is [4 5 6 8 6]. This is invalid schedule. In order to convert it as valid schedule and in this algorithm to explore the population diversity, permutation based trimming [24] is employed. On applying permutation based trimming, resultant schedule will be [4 5 1 3 2]. If there are more than one random key values are same in a schedule (chromosome) then the task comes first is denoted using smaller number. Procedure for evaluation of population is as follows.

Procedure objectivefunct()
1 Begin
2 For each chromosome of the population
3 fitval $\leftarrow 0$
4 For each processor
5 dispatcher (j) $\leftarrow$ task allocated to processor j
6 End for
7 For each processor
8 If (!empty(dispatcher(j)))
9 Compute the fitness value using the function **fitnessfunct**()
10 Update the fitness value fitval
11 Update the processor status
12 Add task to the scheduled list
13 End if
14 End for
15 End

Procedure fittnessfunct()
1 Begin
2 Sort the tasks in dispatcher according to increasing order of its deadline and computation time
3 For all tasks in the dispatcher
4 Assign the ready time R_i, computation time C_i, deadline D_i and start time St_i of each task i
5 Evaluate finish time of the processor i as $Ft_i \leftarrow St_i + C_i$
6 If (($R_i \leqslant St_i$) *and* ($St_i \leqslant D_i - C_i$) *and* (($R_i + C_i$) $\leqslant Ft_i$)) *and* ($Ft_i \leqslant D_i$))7fval $\leftarrow$ *fval* + 1
8 Set the new start time for next process as computation time of the current process i as $St_{next} \leftarrow C_i$
9 End if
10 End for
11 End

Initially, relying on the *arrival time* (A_i) the tasks are sorted in the task queue and kept in safe custody ready queue for processing. For an example, the set of m real-time tasks $\{T_1, T_2, \ldots, T_m\}$ are chosen from the task queue and if the *arrival time* of each real-time task is less than or equal to the current time T_i. $A_i \leq t$ (t is the current time). However, the real-time tasks arrived in the ready queue, are sorted in non-decreasing order of their deadlines and computation times in order to incorporate the popular heuristic earliest deadline first (EDF) and Shortest Computation First (SCTF)

respectively such that tasks do not missed their deadline. Initial population is generated using the first task randomly chosen from the ready queue and the size m is determined based on two conditions. Firstly, if the number of task in the ready queue is more than the number of processors assumed for the multiprocessor environment then $m = no.\ of\ processors$. Otherwise, m is no of task arrived in the ready queue. Here each chromosome of size m is generated randomly by allocating each task in the set $\{T_1, T_2, \ldots, T_m\}$ to a processor (chromosome is represented using an array where each index signifies processor *id* and the value in that index denotes the task assigned to it). Each chromosome in the population is then passed through the fitness evaluation. This helps in determining the number of tasks in each chromosome that meet their deadlines. The chromosomes in the population are then sorted based on the non-increasing order of their fitness values. The first chromosome in the population is marked as the best chromosome. To obtain new population, rotation gates are operated on qubits of the quantum population with the best chromosome C until a maximum number of iterations have been completed or the scheduling criteria is satisfied. When applying rotation gates on each chromosome of the population, new quantum chromosome is reproduced. In each evolution, evaluation of the chromosomes and sorting of the chromosomes based on fitness value obtained. After several iterations the best chromosome that satisfies the scheduling criteria is obtained and the task allotted to each processor is placed on its dispatcher. Each processor then takes the task for execution from its dispatch queue. The real-time tasks allocation using HQIGA are shown in Figs. 6–8.

9. Simulation results

In the simulations, HQIGA algorithm has been implemented on real-time applications. Each real-time task is characterized [28]. Arrival time (A_i), ready time (R_i), maximum computation time (C_i) and deadline (D_i) are the main characteristics for each task T_i in real-time environment. In this work, all these parameters are formulated as follows:

$$A_i = \lfloor (i * 0.05) \rfloor, i = 1, 2, 3, \ldots, m$$

where m is the number of processor

$$R_i = random\ \ numbers\ \ between\ \ [0, 4 * m]$$

$$C_i = random\ \ numbers\ \ between\ \ [\min_C, \max_C]$$

where min_C and max_C are minimum and maximum computation time

$$D_i = random\ \ numbers\ \ between\ \ [R_i + 2 * C_i, R_i + 3 * C_i]$$

In this work, rigorous test has been performed to show the efficacy of the proposed algorithm. The experiments are carried out using randomly generated real-time tasks of 100, 200, 300, 400 and 600. These randomly generated tasks are assigned to number of processors with varying between 10 and 20 processors (variable chromosome size) with population size 30. In order to schedule the tasks according to their initial fitness value using earliest deadline first (EDF) and Shortest Computation Time First (SCTF), tasks are ordered with respect to their deadlines in ascending order.

HQIGA incorporates rotation gate is used to upgrade the qubits in every iteration. The proposed HQIGA algorithm which mimics the basic QIGA algorithm, does not use any basic genetic updating operators like crossover and mutation. Rotation gate operation is relatively faster than the basic genetic operations resulted in less number of iterations required for successful schedule. However, choosing suitable rotation angle confirms the convergence

Table 1
Selection of rotation angle using Look up table.

y_i	b_i	$f(y) \geq f(b)$	$\Delta\omega_i$	$x(\alpha_i, \beta_i)$			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	F	0	0	0	0	0
0	0	T	0	0	0	0	0
0	1	F	0.01π	0	0	0	0
0	1	T	0.0025π	1	+1	±1	0
1	0	F	-0.01π	−1	+1	±1	0
1	0	T	0.025π	+1	−1	0	±1
1	1	F	0.02π	+1	−1	0	±1
1	1	T	0.05π	+1	−1	0	±1

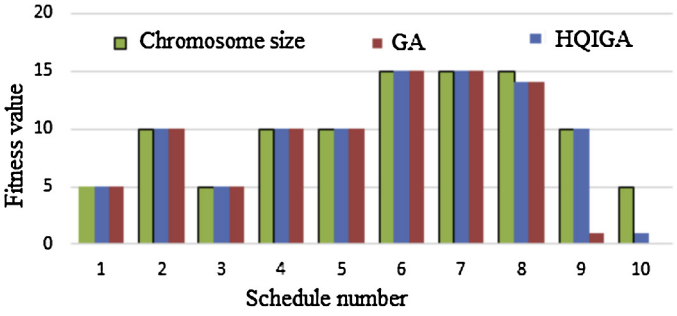


Fig. 3. Comparison of fitness values between HQIGA and CGA in various schedule using EDF.

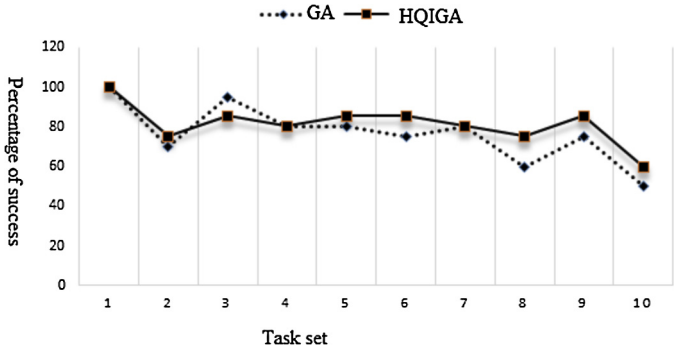


Fig. 4. Percentage of success obtained using fixed chromosome size = 20 using EDF.

of HQIGA. The rotation angle ω is decided as $x(\alpha_i, \beta_i)\Delta\omega_i$, where $x(\alpha_i, \beta_i)$ govern the direction of $\Delta\omega_i$. The selection $\Delta\omega_i$ is shown in Table 1 [54]. The ith bits of the best schedule solution and binary schedule are y_i and b_i respectively.

The current real-time scheduling work have been implemented using classical genetic algorithm (CGA), Hybrid particle Swarm Optimization (HPSO) [9] and proposed Hybrid quantum inspired genetic algorithm (HQIGA) given the maximum permissible limit to evolve is 100 generation. In CGA based approached, the probability for crossover and mutation operation is kept low as 1.0 and 0.1–0.2 respectively considering size of population as 30. Given the variation in chromosome size, comparison of fitness values for both algorithms are shown in Fig. 3 and also the percentage of success is shown in Figs. 4 and 5 with fixed and variable size chromosomes respectively. The percentage of success [23] can be calculated as follows.

$$Percentage\ \ of\ \ success = \frac{number\ \ of\ \ scheduled\ \ task}{total\ \ number\ \ of\ \ task} \times 100\% \quad (15)$$

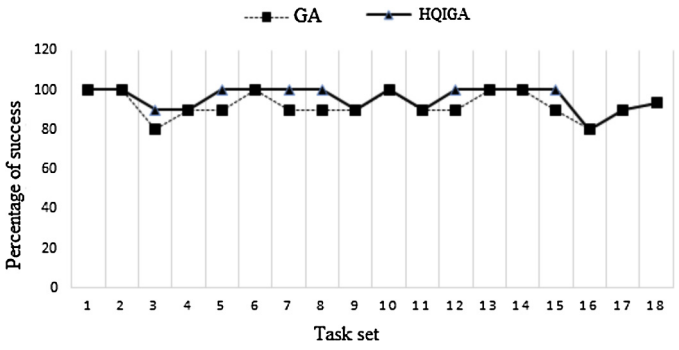


Fig. 5. Percentage of success obtained using variable size chromosome using EDF.

Table 2
Comparative results of HQIGA, CGA and HPSO on the task set = 100 using EDF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	96	97	98	98	99	97.6
	CGA	96	97	97	98	98	97.2
	HPSO	93	89	89	84	78	86.6
No. of Generations	HQIGA	19	19	21	19	21	19.8
	CGA	23	20	21	22	23	21.8
	HPSO	21	20	22	21	20	20.8
Time (s)	HQIGA	18	23	23	24	23	22.2
	CGA	20	23	25	29	20	23.4
	HPSO	22	24	24	27	23	24.0

Table 3
Comparative results of HQIGA, CGA and HPSO on the task set = 200 using EDF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	90	90	89.5	89.5	90	89.8
	CGA	89.5	90	90	89.5	89.5	89.3
	HPSO	88	87	89	78	84	85.2
No. of Generations	HQIGA	44	44	42	43	41	42.8
	CGA	43	48	44	41	44	44.0
	HPSO	45	44	49	40	43	44.2
Time (s)	HQIGA	87	73	88	83	87	83.6
	CGA	87	76	87	86	85	84.2
	HPSO	89	86	82	89	94	88.0

In Figs. 4 and 5, the graphs shows that HQIGA is superior to GA by obtaining 80% and above percentage of feasible task. In addition to this, it is notable fact that the percentage of success in terms of scheduled task using variable size chromosome (95% and above) outperforms the fixed size chromosome (80%) due to starvation caused by fixed number of processors (represented as fixed size chromosome).

In this work, to show the effectiveness of HQIGA algorithm, result is presented on various task sets by varying the number of processors ranging from 1 to 10 (variable chromosome size) and rigorous number of test have been performed and HQIGA. The experimental data in Tables 2–6 show that the HQIGA dominate the solutions by means of meeting deadlines than classical GA based approach and hybrid particle swarm optimization in terms of number of generations required using EDF. In addition, in order to show the effectiveness of EDF over SCTF, experiments are also carried out using SCTF on 100, 200, 300, 400 and 600 data set separately, reported in Tables 7–11. The task set meeting their deadlines generated directly from MATLAB platform are shown in Figs. 6–8 for only 100 task set. For the sake of clarity for all task set are not shown in this paper.

The comparative results are also shown using graphs in Figs. 9–13.

Table 4
Comparative results of HQIGA, CGA and HPSO on the task set = 300 using EDF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	91.6	92	92	92	91.3	91.78
	CGA	92	91	91.6	91.6	91.3	91.5
	HPSO	89	79	87	88	73	83.2
No. of Generations	HQIGA	46	44	48	45	49	46.4
	CGA	48	47	48	46	45	46.8
	HPSO	46	49	47	49	43	46.8
Time (s)	HQIGA	131	121	135	128	132	129.4
	CGA	147	147	159	136	186	155.0
	HPSO	111	145	157	119	143	135.0

Table 5
Comparative results of HQIGA, CGA and HPSO on the task set = 400 using EDF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	85.5	86	85	84.5	85	85.2
	CGA	84	84.5	84	84.5	84	84.2
	HPSO	76	78	77	69	63	72.6
No. of Generations	HQIGA	56	57	50	52	56	54.2
	CGA	60	60	60	60	60	60.0
	HPSO	62	59	59	61	53	58.8
Time (s)	HQIGA	256	271	225	261	219	246.4
	CGA	285	289	245	279	235	266.6
	HPSO	278	289	246	266	226	261.0

Table 6
Comparative results of HQIGA, CGA and HPSO on the task set = 600 using EDF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	85	85.5	87	87	85.7	86.04
	CGA	83.6	86	85	86.5	82.6	84.74
	HPSO	81	87	83	89	73	82.6
No. of Generations	HQIGA	60	60	60	60	60	60.0
	CGA	60	60	60	60	60	60.0
	HPSO	66	68	67	59	63	64.6
Time (s)	HQIGA	323	356	323	345	345	338.4
	CGA	354	321	341	354	334	340.8
	HPSO	355	357	328	319	331	338.0

Table 7
Comparative results of HQIGA, CGA and HPSO on the task set = 100 using SCTF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	84	87	81	78	79	81.8
	CGA	79	75	77	79	77	77.4
	HPSO	81	76	73	79	73	76.4
No. of Generations	HQIGA	17	16	18	19	15	17.0
	CGA	19	18	23	22	21	20.6
	HPSO	16	18	17	19	23	18.6
Time (s)	HQIGA	32	33	38	47	46	39.2
	CGA	47	52	35	45	34	42.6
	HPSO	35	37	28	31	39	34.0

Table 8
Comparative results of HQIGA, CGA and HPSO on the task set = 200 using SCTF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	79	82	75	76	83	79.0
	CGA	75	71	79	73	70	73.6
	HPSO	71	77	73	79	69	73.8
No. of Generations	HQIGA	29	21	29	30	28	27.4
	CGA	30	32	31	33	32	31.6
	HPSO	36	28	27	29	33	30.6
Time (s)	HQIGA	89	80	66	83	67	77.0
	CGA	149	110	134	170	155	143.6
	HPSO	135	157	128	119	131	134.0

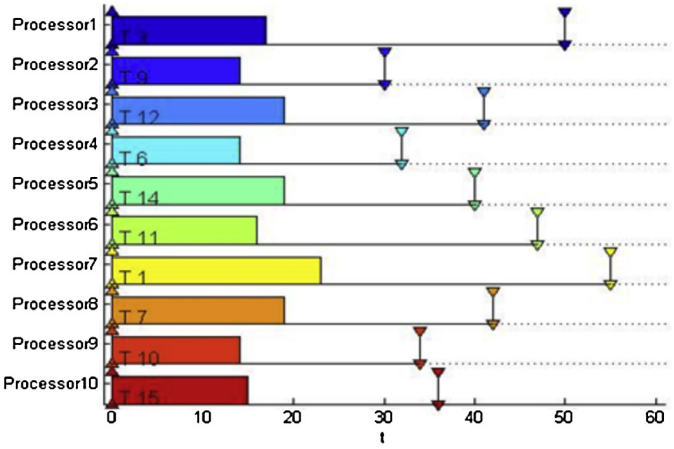


Fig. 6. Schedule chart for first task set where the *X* axis denotes *time* and *Y* axis corresponds for processors using EDF.

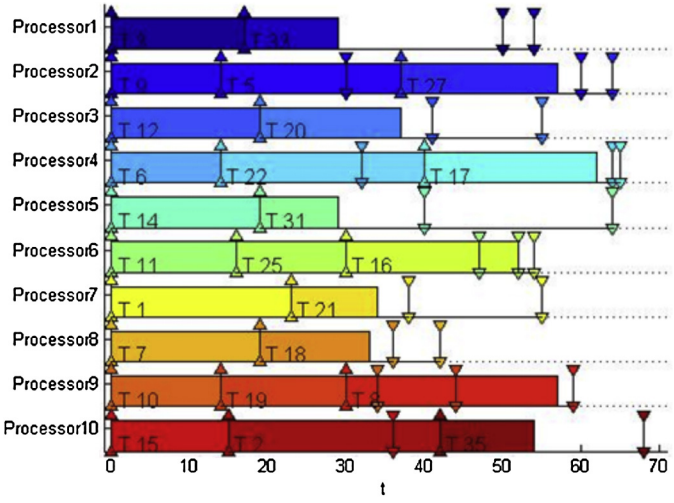


Fig. 7. Schedule chart for intermediate task set where the *X* axis denotes *time* and *Y* axis corresponds for processors using EDF.

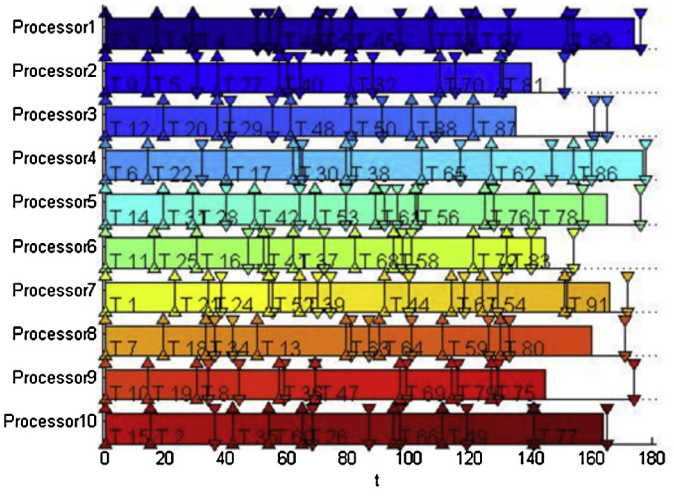


Fig. 8. Schedule chart for final task set where the *X* axis denotes *time* and *Y* axis corresponds for processors using EDF.

Table 9
Comparative results of HQIGA, CGA and HPSO on the task set = 300 using SCTF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	65	65	68	62	67	65.4
	CGA	57	59	61	60	58	59.0
	HPSO	51	57	63	59	53	56.6
No. of Generations	HQIGA	42	41	39	41	42	41.0
	CGA	51	50	52	49	52	50.8
	HPSO	46	41	43	42	53	45.0
Time (s)	HQIGA	112	108	168	130	114	126.4
	CGA	190	162	198	197	187	186.8
	HPSO	159	158	129	179	136	152.2

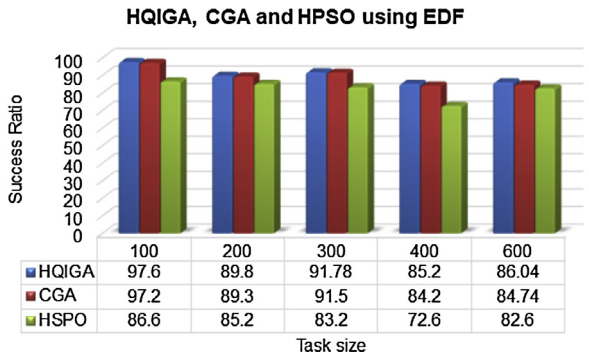
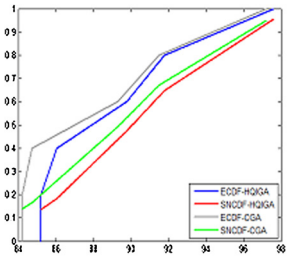
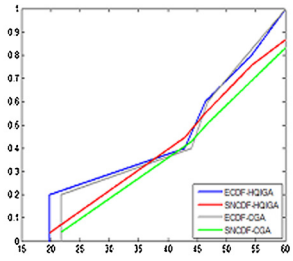


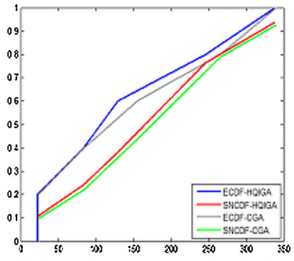
Fig. 9. Comparative results between HQIGA, CGA and HPSO using EDF.



(a) The plot shows the difference between the *empirical cdf* of the HQIGA and CGA on *% of success* and the *cdf* of the *standard normal distribution*



(b) The plot shows the difference between the *empirical cdf* of the HQIGA and CGA on *No. of generations* and the *cdf* of the *standard normal distribution*

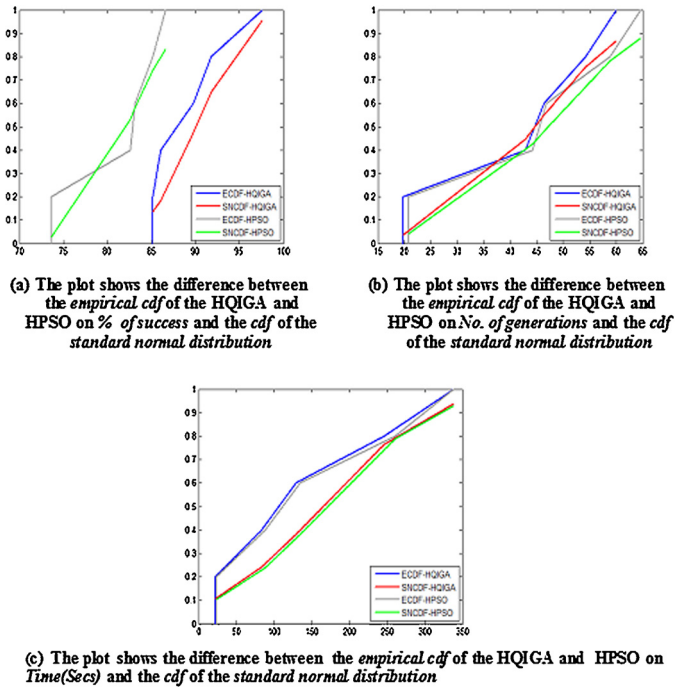


(c) The plot shows the difference between the *empirical cdf* of the HQIGA and CGA on *Time(Secs)* and the *cdf* of the *standard normal distribution*

Fig. 10. (a), (b) and (c) are the graphical comparison for *% of Success*, *No. of Generations* and *Time (s)* considering empirical cdf (ECDF) and standard normal cdf (SNCDF) between HQIGA and CGA (the data and cumulative probability are represented by *X* and *Y* axis respectively).

9.1. Kolmogorov–Smirnov test

If the distribution of sample data is not known in prior and non-parametric in behavior, then one sided Kolmogorov–Smirnov test (KS-test) is preferred for statistical significance test. Moreover, in KS test, two different sample data sets are best described by Empirical Cumulative Distribution Functions (ECDF) while comparative



(a) The plot shows the difference between the *empirical cdf* of the HQIGA and HPSO on *% of success* and the *cdf* of the *standard normal distribution*

(b) The plot shows the difference between the *empirical cdf* of the HQIGA and HPSO on *No. of generations* and the *cdf* of the *standard normal distribution*

(c) The plot shows the difference between the *empirical cdf* of the HQIGA and HPSO on *Time(Secs)* and the *cdf* of the *standard normal distribution*

Fig. 11. (a), (b) and (c) are the graphical comparison for *pcc* and *time* considering empirical cdf (ECDF) and standard normal cdf (SNCDF) between HQIGA and HPSO (the data and cumulative probability are represented by *X* and *Y* axis respectively).

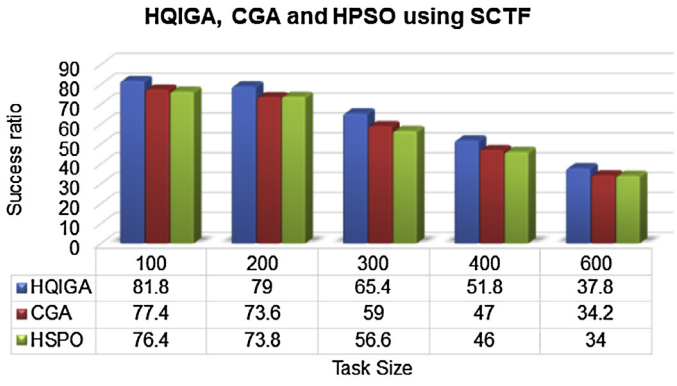


Fig. 12. Comparative results between HQIGA, CGA and HPSO using SCTF.

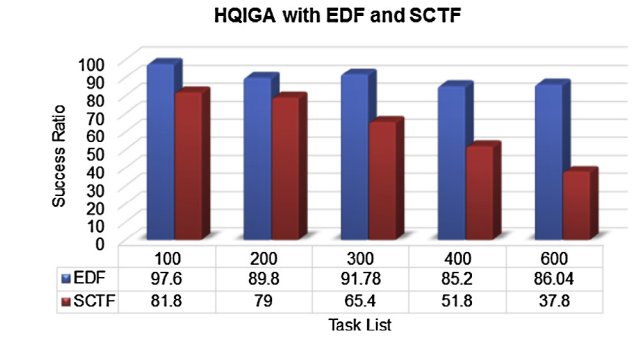


Fig. 13. Comparative results between EDF and SCTF on HQIGA.

analysis is performed. In order to find statistical dissimilarity between two data sets *X* and *Y*, one sided Kolmogorov–Smirnov test is chosen [55]. However, two hypothesis plays significant role in KS: null hypothesis Hyp_0 and alternative hypothesis Hyp_t. Null hypothesis is defined as: if two samples *X* and *Y* are drawn from same distribution or they are equal statistically ($X = Y$). Conversely,

Table 10
Comparative results of HQIGA, CGA and HPSO on the task set = 400 using SCTF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	50	51	55	50	53	51.8
	CGA	47	46	45	49	48	47.0
	HPSO	49	48	43	47	43	46.0
No. of Generations	HQIGA	59	59	59	61	59	59.4
	CGA	61	62	62	59	62	61.2
	HPSO	65	61	63	62	53	60.8
Time (s)	HQIGA	197	195	110	185	169	171.2
	CGA	263	265	268	298	215	261.8
	HPSO	259	258	229	279	236	252.2

Table 11
Comparative results of HQIGA, CGA and HPSO on the task set = 600 using SCTF.

Metric	Algorithms	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Average
% of Success	HQIGA	37	40	34	42	36	37.8
	CGA	33	36	36	31	35	34.2
	HPSO	34	33	33	37	33	34.0
No. of Generations	HQIGA	59	61	62	58	66	61.2
	CGA	59	60	61	61	61	60.4
	HPSO	55	51	63	62	63	58.8
Time (s)	HQIGA	225	200	204	254	245	225.6
	CGA	220	276	298	242	245	256.2
	HPSO	232	281	219	237	259	245.6

Table 12
Two sample one sided Kolmogorov–Smirnov test results between HQIGA and CGA and HPSO using EDF.

	CGA	HPSO
% of Success	=	=
No. of Generations	=	=
Time (s)	=	=

the alternative hypothesis Hyp_t can be defined against the null hypothesis as: if the samples *X* and *Y* are statistically dissimilar, then the data sample are distinct. Interested authors may explore more details about KS test in [56,57].

If the statistical distance between two samples, $Dist_{xy} < C_\alpha$, *critical value* [57], then the null hypothesis Hyp_0 is failed to reject. The critical value C_α is found to be 0.860 during experiment, given the significance level $\alpha = 0.05$ for sample size = 5 and it shows that 95% confidence in favor of the experimental results in KS test. In this current experimental setup, KS test has been performed with proposed HQIGA and CGA and HPSO. The comparison in statistical significance using KS test is reported in Table 12.

Most of the statistical significance test failed to yield optimal result if the size of the sample is not suitably large. Table 12 explores statistical performance and it can be concluded that HQIGA offers similar kind of results while compared in terms of *% of Success, No. of Generations and Time (s)*. However, A little investigation reveals that the numerical outcome produced is in sheer contrast to experimental results shown in Tables 2–6.

In addition to the statistical significance test outcome, graphical comparisons reported between the suggested HQIGA and the CGA and HPSO in terms of cumulative distribution and normal distribution of sample outcome and shown in Figs. 10 and 11.

10. Discussions and conclusion

Various hybrid genetic algorithm based approaches have been proposed for the scheduling of real-time tasks in multiprocessor environment. None, however, has been popularly accepted for high stake tests. The approaches have been guided through the use of

classical genetic algorithm, but the inherent drawback of time consuming operations used in classical genetic algorithm have not been explored. The proposed technique presented a time efficient hybrid algorithm aiming for real-time tasks scheduling by introducing EDF to obviate the classical genetic operators. To produce optimal schedule, the suggested HQIGA relies on the *qubits* for diversified mating pool. Efficacy of the proposed work is achieved by adding values in to the fitness function with the usage of equal number of iterations as classical counterpart.

The problem under consideration is a subset of the much larger field of real-time task scheduling by virtue of the limitation it imposes on the deadlines, preemption and priority. Moreover, proposed Hybrid Quantum-Inspired Genetic Algorithm based approach remains to investigate the performance in the application of multi-objective scheduling for preemptive tasks with priority. Currently, the authors are researching in this direction.

References

[1] I. Ahmad, Y.K. Kwok, A parallel approach for multiprocessor scheduling, in: Proceedings of the 9th International Parallel Processing Symposium (IPPS '95). 1995 IEEE, 1995, doi:1063-7133/95 $10.00.

[2] K. Ramaritham, J.A. Stankovic, P.F. Shiah, Efficient scheduling algorithms for real-time multiprocessor systems, IEEE Transactions on Parallel and Distributed Systems 1 (2) (1990) 184–193.

[3] R.C. Correa, A. Ferreira, P. Rebreyend, Scheduling multiprocessor tasks with genetic algorithms, IEEE Trans. Parallel Distrib. Syst. 10 (1999) 825–837.

[4] S.W. Annie, H. Yu, An incremental genetic algorithm approach to multiprocessor scheduling, IEEE Trans. Parallel Distrib. Syst. 15 (9) (2004) 824–834.

[5] M.R. Bonyadi, M.E. Moghaddam, A bipartite genetic algorithm for multi-processor task scheduling, Int. J. Parallel Progr. 37 (5) (2009) 462–487.

[6] V.M. Nezhad, H.M. Gader, E. Efimov, A novel hybrid algorithm for task graph scheduling, Int. J. Comput. Sci. Issues 8 (2011).

[7] S.N. Sivanandam, P. Visalakshi, A. Bhuvaneswari, Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia, Int. J. Comput. Sci. Appl. 4 (2007) 95–106.

[8] H. Chen, A.K. Cheng, Applying ant colony optimization to the partitioned scheduling problem for heterogeneous multiprocessors, in: Special Issue IEEE RTAS 2005, Work-in-Progress, vol. 2 (No. 2), 2005, pp. 11–14.

[9] M.F. Ercan, A hybrid particle swarm optimization approach for scheduling flow-shops with multiprocessor tasks, in: Proceedings of the International Conference on Information Science and Security, 2008, pp. 13–16.

[10] S. Dhingra, S.B. Gupta, R. Biswas, Genetic algorithm parameters optimization for bi-criteria multiprocessor task scheduling using design of experiments, World Acad. Sci. Eng. Technol. Int. J. Comput. Control Quantum Inf. Eng. 8 (4) (2014).

[11] S.H. Edwin, S.H. Hou, N. Ansari, H. Ren, A genetic algorithm for multiprocessor scheduling, IEEE Trans. Parallel Distrib. Syst. 10 (8) (1999).

[12] S.C. Cheng, Y.M. Huang, Dynamic real-time scheduling for multi-processor tasks using genetic algorithm, in: Computer Software and Applications Conference (COMPSAC), 2004, pp. 154–161.

[13] M. Bohler, F. Moore, Y. Pan, Improved multiprocessor task scheduling using genetic algorithms, in: Int. FLAIRS Conference, 1999.

[14] E.S.H. Hou, N. Ansari, R. Hong, A genetic algorithm for multiprocessor scheduling, IEEE Trans. Parallel Distrib. Syst. 5 (2) (1994) 113–120.

[15] P.A. Laplante, Real-time Systems Design and Analysis. An Engineer Handbook, IEEE Computer Society, IEEE Press, 1993.

[16] M.R. Gary, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP Completeness, W.H. Freeman and Company, 1979.

[17] M.A. Nielson, I.L. Chung, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[18] P.W. Shor, Quantum Computing, Doc. Mathematica. Extra Volume ICM, 1998, pp. 467–486, Available at: http://east.camel.math.ca/EMIS/journals/DMJDMV/xvol-icm/00/Shor.MAN.html.

[19] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proc. 35th Annu. Symp. Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1994, pp. 124–134.

[20] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proc. 28th ACM Symp. Theory of Computing, 1996, pp. 212–219.

[21] L.K. Grover, Quantum mechanical searching, in: Proc. 1999 Congress on Evolutionary Computation, vol. 3, IEEE Press, Piscataway, NJ, 1999, pp. 2255–2261.

[22] A. Narayan, M. Moore, Quantum-inspired genetic algorithms, in: Proc. IEEE Evolutionary Computation, 1996, pp. 61–66.

[23] D. Konar, K. Sharma, S.R. Pradhan, S. Sharma, An efficient dynamic scheduling algorithm for soft real-time tasks in multiprocessor system using hybrid quantum inspired genetic algorithm, in: Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015, Advances in Intelligent Systems and Computing, 2015, pp. 3–11.

[24] K.H. Han, J.H. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, in: Proc. Congress on Evolutionary Computation, 2000, pp. 1354–1360.

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

[26] K.H. Han, J.H. Kim, A quantum-inspired evolutionary algorithm with a new termination criterion, H gate, and two-phase scheme, IEEE Trans. Evol. Comput. 8 (Apr (2)) (2004) 156–169.

[27] B.B. Li, L. Wang, A hybrid quantum-inspired genetic algorithm for multi-objective flow shop scheduling, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 37 (June (3)) (2002) 576–591.

[28] K. Dahal, A. Hossain, B. Varghese, A. Abraham, A. Xhafa, F.A. Daradoumis, Scheduling in multiprocessor system using genetic algorithms, in: Proc. IEEE Computer Information System and Industrial Management Applications, vol. 7, 2008, pp. 281–286.

[29] P. Visalakshi, S.N. Sivanandam, Dynamic task scheduling with load balancing using hybrid particle swarm optimization, Int. J. Open Probl. Comput. Math. 2 (3) (2009) 475–488.

[30] Y.K. Kwok, I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Comput. Surv. 31 (4) (1999) 406–471.

[31] G. Padmavathi, S.R. Vijayalakshmi, A performance study of GA and LSH in multiprocessor job scheduling, Int. J. Comput. Sci. Issues 7 (1) (2010) 37–42.

[32] B. Kruatrachue, T.G. Lewis, Duplication Scheduling Heuristic, a New Precedence Task Scheduler for Parallel Systems. Technical Report 87-60-3, Oregon State University, 1987.

[33] B.S. Macey, A.Y. Zomaya, A performance evaluation of CP list scheduling heuristics for communication intensive task graphs, in: Proc. Joint 12th Intl Parallel Processing Symposium and Ninth Symposium. Parallel and Distributed Processing, 1998, pp. 538–541.

[34] M.R. Mahmood, H.A. Awadalla, Hybrid algorithm for multiprocessor task scheduling, Int. J. Comput. Sci. Issues 8 (2) (2011).

[35] M. Moore, An accurate parallel genetic algorithm to schedule tasks on a cluster, Parallel Distrib. Syst. 30 (5–6) (2004) 567–583.

[36] A.J. Page, T.J. Naughton, Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing, in: The Proceedings of the 19th International Parallel and Distributed Processing Symposium, IEEE Computer Society, Denver, USA, 2005.

[37] M.B. Abdelhalim, Task assignment for heterogeneous multiprocessors using re-excited particle swarm optimization, in: Proc. Int. Conf. on Computer and Electrical Engineering, 2008, pp. 23–27.

[38] H. Talbi, M. Batouche, A. Draa, A quantum-inspired evolutionary algorithm for multiobjective image segmentation, Int. J. Math. Phys. Eng. Sci. 1 (2007) 109–114.

[39] J. Gu, X. Gu, M. Gu, A novel parallel quantum genetic algorithm for stochastic job shop scheduling, J. Math. Anal. Appl. 355 (2009) 63–81.

[40] J. Gu, X. Gu, M. Gu, A quantum genetic based scheduling algorithm for stochastic flow shop scheduling problem with random breakdown, in: Proceeding of the 17th IFAC World Congress, 2008, pp. 63–68.

[41] L. Wang, H. Wu, F. Tang, D.Z. Zheng, A hybrid quantum-inspired genetic algorithm for flow shop scheduling Lecture Notes in Computer Science, vol. 3645, 2005, pp. 636.

[42] 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.

[43] H. Xing, X. Liu, X. Jin, L. Bai, Y. Ji, A multi-granularity evolution based Quantum Genetic Algorithm for QoS multicast routing problem in WDM networks, Comput. Commun. 32 (2009) 386–393.

[44] Y. Jeong, J. Park, J. Shin, A.K. Lee, A thermal unit commitment approach using an improved quantum evolutionary algorithm, Electr. Power Compon. Syst. 37 (2009) 770–786.

[45] A.K. Al-Othman, F.S. Al-Fares, K.M. EL-Nagger, Power system security constrained economic dispatch using real coded quantum inspired evolution algorithm, Int. J. Electr. Comput. Syst. Eng. 1 (2007) 4–10.

[46] P. Benioff, The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines, J. Stat. Phys. 22 (1980) 563–591.

[47] D. Mcmohan, Quantum Computing Explained, John Wiley & Sons, Inc., Hoboken, NJ, 2008.

[48] R. Mall, Real-Time Systems: Theory and Practice, Pearson Education, 2007.

[49] E. Eggers, Dynamic Scheduling Algorithms in Real-time Multiprocessor Systems. Term Paper, EECS Department, Milwaukee School of Engineering, North Broadway, Milwaukee, WI, USA, 1998–99.

[50] G. Manimaran, C. Siva Ram Murthy, An efficient dynamic scheduling algorithm for multiprocessor real-time systems, IEEE Trans. Parallel Distrib. Syst. 9 (3) (1998) 312–319.

[51] R.P. Feynman, R.B. Leighton, M. Sands, The Feynman Lectures on Physics, vol. 3, Addison-Wesley Publishing Company, MA, 1965.

[52] A. Arulselvan, C.W. Commander, P.M. Pardalos, A random keys based genetic algorithm for the target visitation problem, Adv. Coop. Control Optim. (2007).

[53] A. Ruiz, M.A. Sambola, E. Fernández, M.G.C. Resende, A biased random-key genetic algorithm for the capacitated minimum spanning tree problem, Comput. Oper. Res. 57 (2015) 95–108, http://dx.doi.org/10.1016/j.cor.2014.11.011.

[54] R. Nowotniak, J. Kucharski, GPU-based tuning of quantum-inspired genetic algorithm for a combinational optimization problem, Bull. Pol. Acad. Sci.:

Tech. Sci. 60 (October (2)) (2012) 323–330, http://dx.doi.org/10.2478/v10175-012-0043-4.
[55] P. Dutta, P.D. Majumder, Performance Analysis of Evolutionary Algorithm, Lambert Academic Publishers, 2012, ISBN: 978-3-659-18349-2.
[56] D. Konar, S. Bhattacharyya, B.K. Panigrahi, K. Nagamatsu, A quantum bi-directional self-organizing neural network (QBDSONN) architecture for binary object extraction from a noisy perspective, Appl. Soft Comput. 46 (2016) 731–752.
[57] M.H. Gail, S.B. Green, Critical values for the one-sided two-sample Kolmogorov–Smirnov statistic, J. Am. Stat. Assoc. 71 (355) (1976) 757–760.