Quantum Optimization Based Resource Distribution Management System

Ph. D. Dissertation

by

Sara El Gaily (MSc)

Supervised by

Prof. Dr. Sándor Imre (cmHAS)

Department of Networked Systems and Service

Budapest University of Technology and Economics

Hungary, 2022

Abstract

Resource distribution management (RDM) must be designed to be highly reacting fast with maximum accuracy and speed to any unpredictable workload. As it is known, tasks with a fixed running time require more computation in a real-time system, since they are executed at a constant rate. To rationally use computing resources as a way to reduce the computational complexity of finding the best optimum deployment scenarios under the imposed constraints, quantum computing methods have been utilized to handle this assignment problem. This dissertation provides and summaries the study about the optimization of the RDM system by exploiting quantum optimization algorithms.

First, a set of different quantum optimization algorithms has been discovered and introduced, where each proposed quantum optimization approach is capable of dealing with a particular type of problem optimization. Following that, a study has been presented that shows that these proposed quantum optimization techniques reduce computational complexity while increasing accuracy significantly. Moreover, a classical and quantum uncertainty study of the suggested quantum optimization methods, as well as their computational complexity has been conducted.

Second, to prove the efficiency of the proposed quantum optimization strategies, a generalized model for the RDM system while considering multiple task types has been built. Following that, the Quantum Extreme Value Searching Algorithm (QEVSA) has been implemented as an embedded computational infrastructure in the decision-maker in order to achieve a load balance. Next, a parameter setup of the QEVSA with a consistent computational complexity study according to the given model has been provided. Finally, a simulation environment has been designed to demonstrate the efficiency of the proposed quantum optimization strategy.

Third, the QEVSA has been implemented as an embedded computational method in an extended version of the RDM system while considering multiple tasks and multiple subtasks. The goal was to reach a load balance. Following that, the stochastic parameters of QEVSA have been tuned. Then, we proved that the proposed optimization problem cannot be handle by classical optimization algorithms. Finally, simulation environments show that the quantum optimization strategy attains a high load balance and low computational complexity.

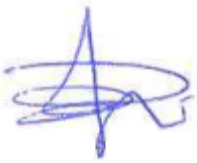
Fourth, the Constrained Quantum Optimization Algorithm (CQOA) has been incorporated in the RDM to minimize the overall energy consumption by taking into consideration the delay constraint of the incoming tasks. Next, an analytical study according to the suggested RDM model has been provided. This is followed by a configuration of the CQOA After, we showed that the computation of the proposed constrained assignment problem is a challenging task and cannot be solved through classical constrained optimization algorithms. Finally, the analytical results have been validated by simulation experiments.

Finally, the CQOA has been exploited to reduce the energy usage of RDM while taking queueing scenarios into account. Furthermore, we demonstrated that the computational complexity of selecting the optimum distribution scenarios within the database is polynomial in terms of the numbers of the computing units but exponential in terms of the numbers of subtasks in case of sharing a free access waiting queue among computing units of the RDM system. Finally, modeling experiments have been used to verify the analytical results.

I, the undersigned Sara El Gaily, hereby state that I have written this doctoral dissertation myself, and I have used only the sources given in it. I have clearly marked all the parts taken from other sources either word for word or reworded but with the same contents, indicating their sources.

The reviews of the dissertation and the report of the thesis discussion are available at the Dean's Office of the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics.

Budapest, March 18, 2022



Sara El Gaily

Acknowledgments

First of all, I would very much like to express my huge and sincere thanks to my professor and supervisor Dr. Sándor Imre for his kind and highly appreciated efforts, patience, help, and extensive support during the last four years of supervision.

Secondly, I would like also to thank all members of the Department of Networked Systems and Services at the Budapest University of Technology and Economics for their beloved support and cooperation during my whole Doctorate career.

Last but not least, I am very grateful to my family, for their support and help.

Content

Acronyms

BSA	Binary Searching Algorithm
CDS	Continuous Database Structure
CQOA	Constrained Quantum Optimization Algorithm
CQRT	Constrained Quantum Relation Testing
IQFT	Inverse Quantum Fourier Transform
MCSA	Minimum Constrained Searching Algorithm
NCDS	Non-Continuous Database Structure
NCT	No-Cloning Theorem
QAOA	Quantum Approximate Optimization Algorithm
QC	Quantum Computing
QCA	Quantum Counting Algorithm
QE	Quantum Entanglement
QET	Quantum Existence Testing
QEVSA	Quantum Extreme Value Searching Algorithm
QMSA	Quantum Minimum Searching Algorithm
QOANCDS	Quantum Optimization Algorithm for Non-Continuous Database Structure
QP	Quantum Parallelism
QPE	Quantum Phase Estimation
RDM	Resource Distribution Management

List of Figures

List of Tables

1 Introduction

Cloud resource demands are rapidly increasing exponentially with the development of cloud computing [1-8]. For this sake, new modern methods and techniques are needed to improve its quality of service. In this work, the problem of resource distribution management (RDM) in cloud computing is formulated as an optimization problem.

This Chapter is organized as follows. First, an overview of the main concepts of quantum computing and its applications has been presented. Next, a study about the advantages and deficiencies of the well-known existing classical optimization methods has been investigated as well as the state of the art of the favored classical optimization strategies that are utilized in cloud computing. Finally, an overview of the forthcoming Chapters has been introduced.

1.1 Quantum computing overview and applications

Quantum computing and communication technologies are going to be exploited in many application domains in the foreseeable future (see [9]-[10]). It is interesting to point out that quantum computing (QC) infringes the fundamental basic concept, called the classical complexity-theoretic Church–Turing thesis, claims that to run a computational problem rapidly, it is necessary to apply one of these solutions, either designing an algorithm with minimum steps, or performing the steps with more computing units in parallel, or decreasing the processing delay of an implemented single step.

The postulates of QC form powerful mathematical tools for modeling a real physical system, as well as, it consolidates a strong understanding of QC [9-10]. Here as follows, we describe briefly these postulates.

1st postulate:

It describes the notion of a "qubit" which is the smallest quantum information in QC, a qubit is a superposition of two computational basis states $|0>$ and $|1>$. For example, a qubit is described as follows, $|\varphi>= a|0> +b|1>$, where a and b refer to the complex probability amplitudes of the two possible computational basis states $|0>$ and $|1>$, such that $|a|^2 + |b|^2 =$

1. The qubit can be in both basis states at the same time, this property allows quantum computers to perform better than classical ones.

2nd postulate:

This postulate introduces the notion of a quantum gate which is nothing else than a unitary operator *U*, i.e. *U* is a unitary operator ($U^{-1} = U^{\dagger}$). It is interesting to note that the evolution of the quantum state depends only on the starting and finishing time. The reason behind using unitary operators as quantum gates is that they conserve the reversibility of the quantum evolution, i.e. if we know the input we get back the output and vice versa, as well as; keep the sum of the power of the probability amplitudes equal to one.

3rd postulate:

This postulate presents the measurement notion in the quantum world which is considered as a bridge between the quantum and the classical world. Performing a measurement means breaking down the property of the quantum behavior. Now, let's assume that the following quantum state $|\gamma> = \sum_{i=1}^{N} \gamma_i \, |i>$. It is interesting to note that measurement can be described through a set of measurement operators $\{M_m\}$ where each of them stands for a possible result *m*, the probability of measuring *m* when the system is in state $|\gamma>$ can be expressed as follows,

$$P(m||\gamma>) = <\gamma| \, {M_m}^{\dagger} M_m \, |\gamma>, \tag{1.1}$$

where ${M_m}^{\dagger}$ is the adjoint of M_m and $<\gamma|$ is the adjoint of $|\gamma>$. As we already discussed, the measurement influences the observed outcome. For this sake, it is necessary to compute theoretically the post-measurement state. The system after measuring *m* gets the state,

$$|\gamma'> = \frac{M_m \, |\gamma>}{\sqrt{<\gamma| \, {M_m}^{\dagger} M_m \, |\gamma>}}. \tag{1.2}$$

The completeness relation presented in Eq. (1.3) provides us information about the correctness of the given constructed measurement device,

$$I = \sum_m M_m{}^{\dagger} M_m. \tag{1.3}$$

4th postulate:

This postulate presents the concept of a quantum register which can be composed using the tensor product of the quantum states. Let's consider the following quantum states $|\gamma_1>$,…, $|\gamma_n>$, such that $n \in \mathbb{N}$, the joint state of the composite is represented as

$$|\gamma> = |\gamma_1> \otimes ... \otimes |\gamma_n>. \tag{1.4}$$

It is worthwhile to mention that some sophisticated and efficient quantum-world phenomena such as Quantum Entanglement (QE) and the No-Cloning Theorem(NCT), are not present in classical computation.

The QE is a mysterious phenomenon in quantum computing, it has no analogous in its classical counterpart. It is a special correlation between two or more quantum states. For example, if two entangled qubits are considered and they are spatially separated, if we measure one of them, we can determine the state of the other qubit. The QE provides new ways of transmitting and processing information and it is frequently exploited in quantum cryptography, quantum communication channels, and quantum teleportation. It is interesting to note that the most known quantum entangled states are the Bell states,

$$|\beta_{ab}\rangle = \frac{|0, b\rangle + (-1)^a |1, NOT(b)\rangle}{\sqrt{2}} \quad a, b \in \{0,1\}. \tag{1.5}$$

The NCT states that it is impossible to clone an unknown quantum state, but we can only make a copy of the known quantum states or orthogonal quantum states. This theorem has a powerful effect and an important role in quantum cryptography because it forbids eavesdroppers from making copies of quantum keys in a quantum channel.

It is important to point out that quantum algorithms are designed to solve computationally complex hard tasks. Any quantum algorithm has four main components:

- Initialization part: preparing a quantum register which consists of a superposition of all the possible states with uniform probability amplitude.

- Quantum Parallelism (QP) is responsible for processing all the possible results in only one step.

- The amplification part is responsible for amplifying the probability amplitude of the wanted result (searched value or item) to converge to 1, this step is performed either by the Hadamard gates or by applying the Inverse of Quantum Fourier Transform (IQFT).

- The measurement where we return to the conventional world by obtaining the result.

Quantum algorithms can offer exponential speedup in solving problems that cannot be handled classically. In the last decades, several well-known quantum strategies have been proposed in QC [9,10] such as quantum phase estimation (QPE) which is exponentially faster than the classical ones, it computes the eigenvalue of a unitary operator, it has many useful applications, here we list some examples, such as, quantum counting algorithm (QCA) for computing the number of occurrences of a query (searched item) in a certain database, Shor's algorithm for integer factorization [11], or the HHL algorithm [12] for solving a linear system of equations. Another interesting well-known searching algorithm, the so-called Grover's algorithm (quantum solution for searching an item in an unsorted database) which enables a dramatic reduction in computational complexity. The optimal classical solution takes $O(N)$ iterations to carry out the search while Grover's strategy requires only $O(\sqrt{N})$ steps [13-14].

1.2 Resource distribution management and optimization methods

The RDM challenges arise due to the need to maximizing the throughput; reducing the power consumption, the cost, the delay, and the computational complexity; reaching a load balance. The complexity degree of solving the optimization problem depends also on the dependencies nature of the adopted attributes defining the elements of the model, the architecture and the size of the system, the type of the metric used. For this purpose, many research has been driven to find solutions to the above-mentioned problem paradigms. Furthermore, there always exists a trade-off between all the challenges mentioned previously. All the challenges that have been discussed about resource distribution management are outlined in Figure.1.1.

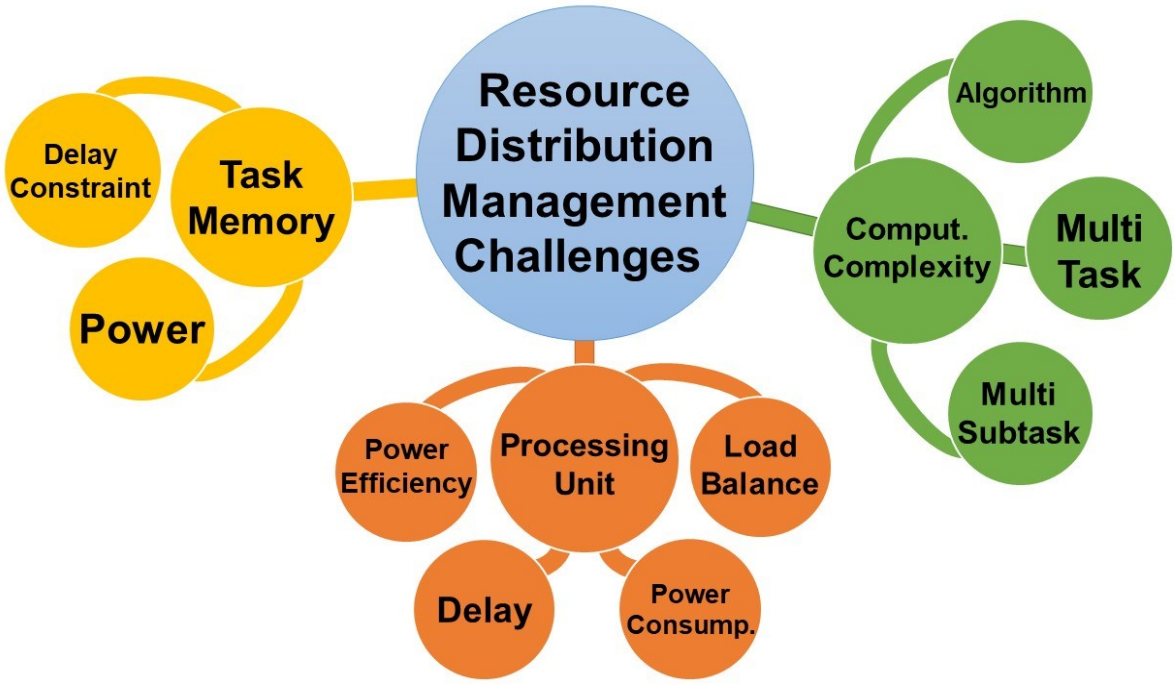


Figure 1.1: Resource distribution management challenges

This study investigated quantum computing strategies in the RDM paradigm. To show the efficiency of the suggested quantum algorithms, reducing the energy consumption and balancing the load were considered as toy metrics. This section introduces the well-known classical and quantum optimization algorithms as well as the most preferred optimization strategies used for reducing energy consumption and balancing the load in a cloud environment.

In compliance with what has been discussed earlier, managing the massive exponential increase of cloud resource demands and improving resource utilization is highly favorable by the service providers. Hence, comprehensive and efficient resource distribution methods that handle both computational complexity and network resources are needed [15-17].

Let us list some most well-known optimization algorithms. First, starting with the quantum world. It is interesting to point out that it is not easy to build a new efficient quantum optimization method that outperforms a classical one. To that end, there are few discovered quantum optimization algorithms. Concerning the extreme value searching, the most leading quantum heuristic candidate is introduced in [18], the so-called quantum approximate optimization

algorithm (QAOA) which approximates hard optimization problems by converting the classical objective function into a Hamiltonian problem. Later, the QAOA was exploited for solving a constrained optimization problem, this alternative solution investigates the ground state of the Hamiltonian cost function instead of minimizing the original cost function [19]. Another extended version of the QAOA, the so-called quantum alternating operator ansatz is designed for finding the approximate solutions to optimization problems with hard constraints [20]. The only drawback is that the calculation of the derivatives of the goal function can be computationally expensive. On the other hand, the most well-known classical optimization algorithms are classified into two categories: the traditional deterministic methods (see [21]) and the metaheuristic ones (see [22]). Among the deficiencies of the deterministic methods is that they often fail to find the global extreme value solution of the goal function (especially, if the database contains many local extreme values) because the function may be non-continuous or non-monotonic. While the metaheuristic methods are the most favored by researchers to handle engineering optimization problems. Nevertheless, these strategies are characterized by high computational complexity due to a large number of goal function evaluations.

Task distribution has been proved as an NP-complete problem [23]. Processing or allocating user requests (tasks) in a cloud environment is considered a crucial optimization issue. Minimizing energy consumption has become a vital issue than ever due to the explosive use of the internet and the increasing size of cloud computing. Also, the problem of load balancing which allows voiding failures concerning execution time and energy consumption has attracted significant attention.

Several studies have been conducted to solve the problem of energy utilization in cloud computing. Though diverse strategies and achievements, the suggested methods are very distinct and minimizing energy consumption is still challenging. The virtualization technology and Dynamic Voltage and Frequency Scaling techniques have been among the efficient solutions for minimizing energy consumption [24]. On the other hand, the deterministic (heuristic) and metaheuristic optimization strategies have also received considerable interest and have been extensively exploited in the cloud environment so that energy is consumed optimally. Some metaheuristic algorithms are often used such as Ant Colony Optimization [25-27], Particle Swarm Optimization [28-29], Genetic Algorithm [30-32], Bacteria Foraging Algorithm [33]. However,

the aforementioned optimization algorithms achieve an efficient and accurate search but they perform a time-consuming computation.

In the literature, there are two types of load balancing algorithms in cloud computing: static and dynamic [34-35]. The static based-balancing strategies are mostly implemented in a stable environment with a homogenous system. The dynamic-based balancing algorithms are more flexible and efficient in both homogeneous and heterogeneous environments. Some well-known static methods are the round-robin load-balancing algorithm, load-balancing min-min algorithm, load-balancing min-max algorithm. Dynamic load balancing has two categories: distributed and centralized. Nevertheless, these methods are characterized by high computational complexity. Also, there is a considerable number of researches that attempt to solve load balancing by using heuristics and metaheuristics algorithms [36-39].

The task assignment problem is a challenging problem due to the following reasons. Firstly, the deployment strategy should take different task types into consideration as well as the tasks' nature (dependent or independent tasks). Secondly, the incoming tasks usually have to be served within different delay constraints. Thirdly, a task may consist of multiple subtask types. Numerous researches focused on optimizing the energy consumption (or balancing the load) of cloud computing and neglecting the subtask level.

1.3 Organization

The dissertation summarizes a study about exploiting quantum computing methods in handling RDM problems, inspired by the need of dealing with computational complexity challenges. Testing and validating the results obtained analytically have relied on extensive simulation environments using Matlab and Python.

First, two new quantum optimization algorithms have been developed, where the classical and quantum uncertainty have been investigated as well as their computational complexity. Next, a comparison between the well-known classical optimization methods and quantum optimization strategy in terms of computational complexity and accuracy has been presented, the results and contributions are presented in Chapter 2.

Second, an unconstrained quantum optimization algorithm has been implemented in a RDM paradigm to attain load balancing and low computational complexity. Next, a configuration of the stochastic parameters of the proposed quantum method with respect to the RDM model has been investigated. Then, numerical results show the efficiency of quantum strategy in terms of load balancing and computational complexity, the results and contributions are presented in Chapter 3.

Third, the RDM paradigm has been enhanced with a more difficult work assignment challenge. Then, an unconstrained quantum optimization technique was applied in the RDM. Following that, an analytical study shows that the proposed task distribution optimization problem cannot be solved by regular computers. Numerical results show that the quantum strategy is efficient in terms of load balancing and computational complexity, the results and contributions are presented in Chapter 4.

Fourth, a constrained quantum optimization algorithm has been implemented in the RDM paradigm to achieve a significant reduction in terms of energy consumption and computational complexity. Following that, a configuration of the constrained quantum method's stochastic parameters has been investigated. Simulation results show that the constrained quantum strategy is efficient in saving energy and reducing computational complexity, the results and contributions are presented in Chapter 5.

Last, a constrained quantum optimization strategy has been exploited to reduce the energy usage of RDM while taking queueing scenarios into account. Next, analytical results show that the computational complexity of the overall possible deployment scenarios is polynomial in terms of the number of computing units and number of tasks waiting in the queue but exponential in terms of the number of subtasks. Finally, a simulation environment has been built. The results and contributions are presented in Chapter 6.

2 Quantum optimization algorithms

In engineering and science domains, solving optimization problems has a wide range of applications. Optimization algorithms aim to find the best extreme (maximum or minimum) outcome of an objective function or an unsorted database. It is worth noting that these optimization methods can be divided into two categories: unconstrained and constrained optimization methods. The majority of existing problems are computationally challenging. Quantum optimization algorithms, unlike their classical counterparts, use the principles of quantum mechanics to speed up computations.

2.1 Introduction

The present Chapter introduces two new quantum optimization algorithms:

- The first one is inspired by the quantum extreme value searching algorithm (QEVSA) [40-41] which finds the extreme optimum result of an unconstrained goal function or unsorted database where the type of the database structure can be continuous or non-continuous (The definition of the continuous and non-continuous databases are going to be provided later), this method labeled Quantum Optimization Algorithm for a Non-Continuous Database Structure (QOANCDS). It is worth mentioning that the unsorted databases can be regarded as goal functions with discrete input such that the database entries can be associated with numbers.
- The second method is an extension of the QOANCDS, it searches the extreme value of a constraint goal function, This quantum strategy named Constrained Quantum Optimization Algorithm (CQOA).

Later, we will show how each quantum optimization algorithm exceeds the other one. In the next chapters, these new quantum optimization algorithms will be considered as a computational infrastructure for RDM system**.**

The rest of the Chapter is organized as follows. Section 2.2 presents the necessary background for developing the QOANCDS and CQOA. Section 2.3 introduces the QOANCDS and investigates its classical and quantum uncertainty as well as its computational complexity. Section 2.4 presents the CQOA and investigates its classical and quantum uncertainty as well as its computational complexity. Section 2.5 discusses the computational complexity of the QEVSA, CQOA, and QOANCDS and highlights the difference between them. An analytical comparison between the proposed QEVSA and the conventional heuristic optimization algorithms is presented in Section 2.6, while Section 2.7 is a conclusion of the chapter section.

2.2 Necessary background for developing the CQOA and QOANCDS

Before developing the QOANCDS and CQOA, certain key concepts such as QPE, Grover's algorithm, QCA, continuous and non-continuous database structure, and QEVSA must be reviewed.

As we already discussed in Chapter 1, the smallest information in quantum computing is the qubit $|\varphi> = a|0> + b|1>$, where $|0>$ and $|1>$ are the computational basis states such that,

$$|a|^2 + |b|^2 = 1 \qquad a, b \in \mathbb{C}. \tag{2.1}$$

Eq. (2.1) has two implications,

- Operations on the quantum state must conserve the norm of the output state, for this purpose, the quantum gate is a unitary matrix U. The output state can be calculated as $|\gamma> = U|\varphi>$.

- The eigenvalues and the eigenvectors of the unitary matrix U can be written as $U|x> = e^{i\Omega}|x>$, where $e^{i\Omega}$ is the eigenvalue of U and Ω is a real number.

The QPE extracts with an exponential speed up the phase Ω of the eigenvalue of a unitary operator. Additional analyses are discussed in [9-10] on the operating process and physical implementation of QPE.

Three main blocks constitute any quantum algorithm, the first block is the initialization block where all of the possible outcomes are encapsulated in one quantum register, the second block

processes the input according to the goal result, while the third block amplifies the obtained result to be ready for successful measurement.

Grover's algorithm finds the given query (searched item) in an unsorted database with $\sqrt{N}$ steps, where N is the size of the database. Grover's algorithm comprises of two steps, in the first step all the possible outcome states are prepared with another auxiliary qubit in a single quantum register called $|A>$, in the second step, Grover's operator G is applied, this G operator works simultaneously as a processing box for inputs and amplifier of the wanted outcome, more details are given in [9-10]. Suppose that the number of occurrences of the searched item in the database denoted by M. The expression of $|A>$ can be expressed as follows,

$$|A> = \sqrt{\frac{N-M}{N}}|\alpha> + \sqrt{\frac{M}{N}}|\beta>, \tag{2.2}$$

where $|\alpha>$ and $|\beta>$ are two orthonormal basis referring respectively to the unwanted and the wanted states, and M refers to the number of occurrences of the searched in the database. The geometrical interpretation on the unit circle of Grover's operator is as follows: $|\alpha>$ and $|\beta>$ represent respectively the horizontal and vertical basis, the $|A>$ forms with $|\alpha>$ an angle equal to $\frac{\Omega}{2}$, The operator G is nothing else than the processing and amplification operations in disguise, it conducts to rotate the initial state such that it became closer to $|\beta>$. For more review about the geometrical representation of the Grover operator, see [9-10]. The operator G must be applied for an optimal number of times L_{opt} to find the searched item, where $L_{opt} \cong \frac{\pi}{4}\sqrt{\frac{N}{M}}$.

Grover's algorithm cannot compute the value of M. To that end, the QCA comes into the picture to handle the problem of computing M. The surprising news is that QPE works in a disguised way for determining M because the value of M is connected to the phase Ω of the eigenvalue of Grover's operator which is equal to $e^{\pm i\Omega}$ using Eq. (2.3), more details are given in [9-10].

$$\frac{\Omega}{2} \cong \sqrt{\frac{M}{N}}. \tag{2.3}$$

To understand the similarities and the differences between the QEVSA and the new extended quantum optimization algorithms (QOANCDS and CQOA) explained in the next subsections 2.3 and 2.4, it is necessary to put the reader into the picture by firstly presenting the definition of a continuous database structure (CDS), then, introducing the QEVSA.

The QEVSA is a stochastic process working on an unsorted database, it combines the well-known classical Binary Searching Algorithm (BSA) [42] and Quantum Existence Testing (QET) method [40-41]. The QEVSA finds the optimum extreme (minimum or maximum) value of an unconstrained objective function.

The QET is a special case of the QCA, it checks whether the searched item exists or not in the database, i.e. The QET answers YES if the searched item exists in the database or NO if it does not. It is not required to compute the exact value *M*. The QET investigates whether the initial quantum state $|A>$ forms an angle different than zero or not with the basis state $|\alpha>$ if the calculated angle equals zero, it means that the searched item does not exist in the database, else it does. The power of QET is derived from the QPE which keeps the efficiency of the BSA while processing the search in an unsorted database.

The QEVSA is introduced in detail as follows:

---

1. We start with $S = 0$: $T_{min1} = T_{min0}$, $T_{max1} = T_{max0}$, and $\Delta T = T_{max0} - T_{min0}$
2. $S = S + 1$
3. $T_{med\,S} = T_{minS} + \left[\frac{T_{maxS} - T_{minS}}{2}\right]$
4. $flag = QET\ (T_{med\,S})$:
    - $flag = Yes,$ then $T_{max\,S+1} = T_{med\,S}$, $T_{\min S+1} = T_{\min S}$

- Else $T_{max\,S+1} = T_{max\,S}$, $T_{min\,S+1} = T_{med\,S}$

5. If $S < log_2\,(T)$, then go to 2, else stop and $y_{opt} = T_{med\,S}$

---

where T refers to the maximum number of steps needed to run the BSA embedded in the QEVSA. This stochastic parameter has an important role in running the QEVSA.

The computational complexity of the QEVSA depends on,

- The computational complexity of the BSA embedded in the QEVSA which equals $O(log_2(T))$.
- And, the computational complexity of the QET which equals $O\left({log_2}^3(\sqrt{N})\right)$, where $N = 2^a$ is the entry size of the database, where a is the total number of the required qubits with respect to the size N of the database. Figure 2.1 shows the physical implementation of the QET.

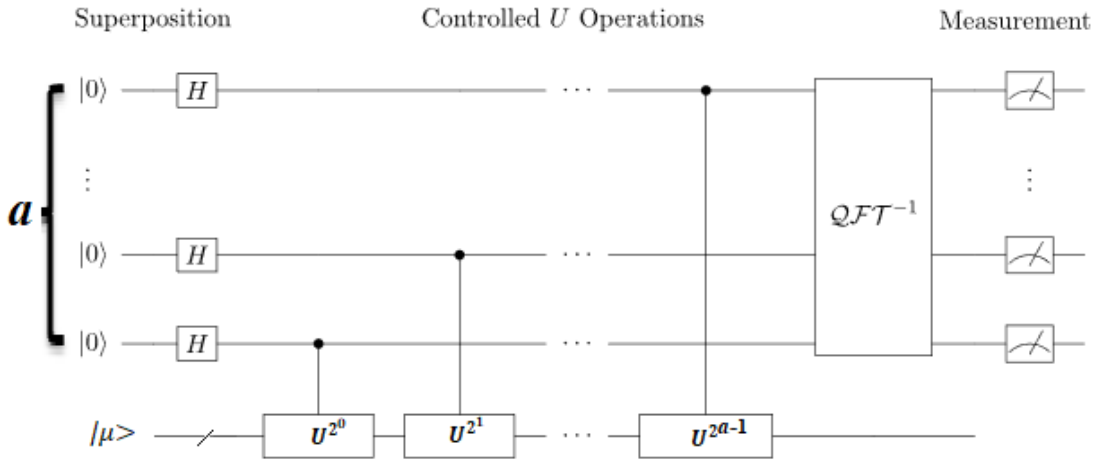


Figure 2.1: Quantum existence testing device.

The overall number of bits n_E used in the physical implementation of the QET is strongly influenced by the quantum uncertainty and the classical accuracy of the application. In case the

quantum uncertainty demand is neglected, i.e. it corresponds to the idealistic phase estimation with no error, this implies that the value of n_E can be written as

$$n_E = \underbrace{\frac{a}{2} - 1}_{c_E}, \tag{2.4}$$

where c_E is the optimum number of qubits required for classical accuracy in order to represent the phase, in this case, the computational complexity of the QET is $O\left(log_2{}^3\left(\sqrt{N}\right)\right)$. On the other hand, if the upper bound of the error probability denoted by $\breve{P}_\varepsilon$ of the quantum uncertainty _originated from the QPE_ is taken into consideration by the application, then the value of n_E is expressed as

$$n_E = \underbrace{\frac{a}{2} - 1}_{c_E} + \underbrace{\left\lceil log_2(2\pi) + log_2\left(\frac{1}{8\breve{P}_\varepsilon}\right)\right\rceil}_{p_E}. \tag{2.5}$$

where p_E is the optimum number of qubits needed to handle the quantum uncertainty problem originating from the error probability of converting the phase to a probability amplitude. In this respect, the computational complexity of the QET is $O\left(log_2{}^3\left(2^{\left(\frac{p_E}{2}\right)^2}\sqrt{N}\right)\right)$. We assume that the computational complexity of the QEVSA is only influenced by the classical certainty parameter. For this sake, the computational complexity of QEVSA can be written as $O\left(log_2(T)log_2{}^3\left(\sqrt{N}\right)\right)$. In compliance with what has been discussed, we see that the QET plays a fundamental role in the search efficiency of the QEVSA.

Now, let's present the definition of the CDS. For every index register *x* that has a value in the database denoted by $DB[x]$, the index registers take values from 0 to *N*-1, while for every index register *x* the value of $DB[x]$ is a real value. Let's represent the index register on the *x*-axis, where $x \in [0, N-1]$. As an explicative example, the index registers and their database values are shown in Figure 2.2.a. It is worth mentioning that the scaling of the *y*-axis relies strongly on the values of the database such that the smallest absolute difference between $DB[x]$ and $DB[y]$ is denoted by α, one writes

$$\min_{\forall x,y\in[0,N-1]} |DB[x] - DB[y]| = \alpha. \quad (2.6)$$

The unit scale of the *y*-axis equals α, instead of using this original unit scaling, we convert the database values of the *y*-axis from the original unit scale (The unit scale of the *y*-axis equals α) into a new unit scale which equals 1 (the value of α is substituted by 1). Figure 2.2.b and Figure 2.2.c show the conversion from the original unit scale into the new unit scale of the *y*-axis. A detailed discussion on how to set up α is given will be given in Chapters 3 and 4.

A database structure is being continuous if for every two neighboring index registers *x* and *x*+1 the absolute value of $|DB[x+1] - DB[x]|$ can be expressed as

$$|DB[y+1] - DB[y]| = 1 \; or \; 0. \quad (2.7)$$

Figure 2.2.c illustrates the CDS. The QEVSA can only handle the search optimization for the CDS. A schematic representation of the QEVSA as minimum searching algorithm (QMSA) example is illustrated in Figure 2.3 where its steps are described.

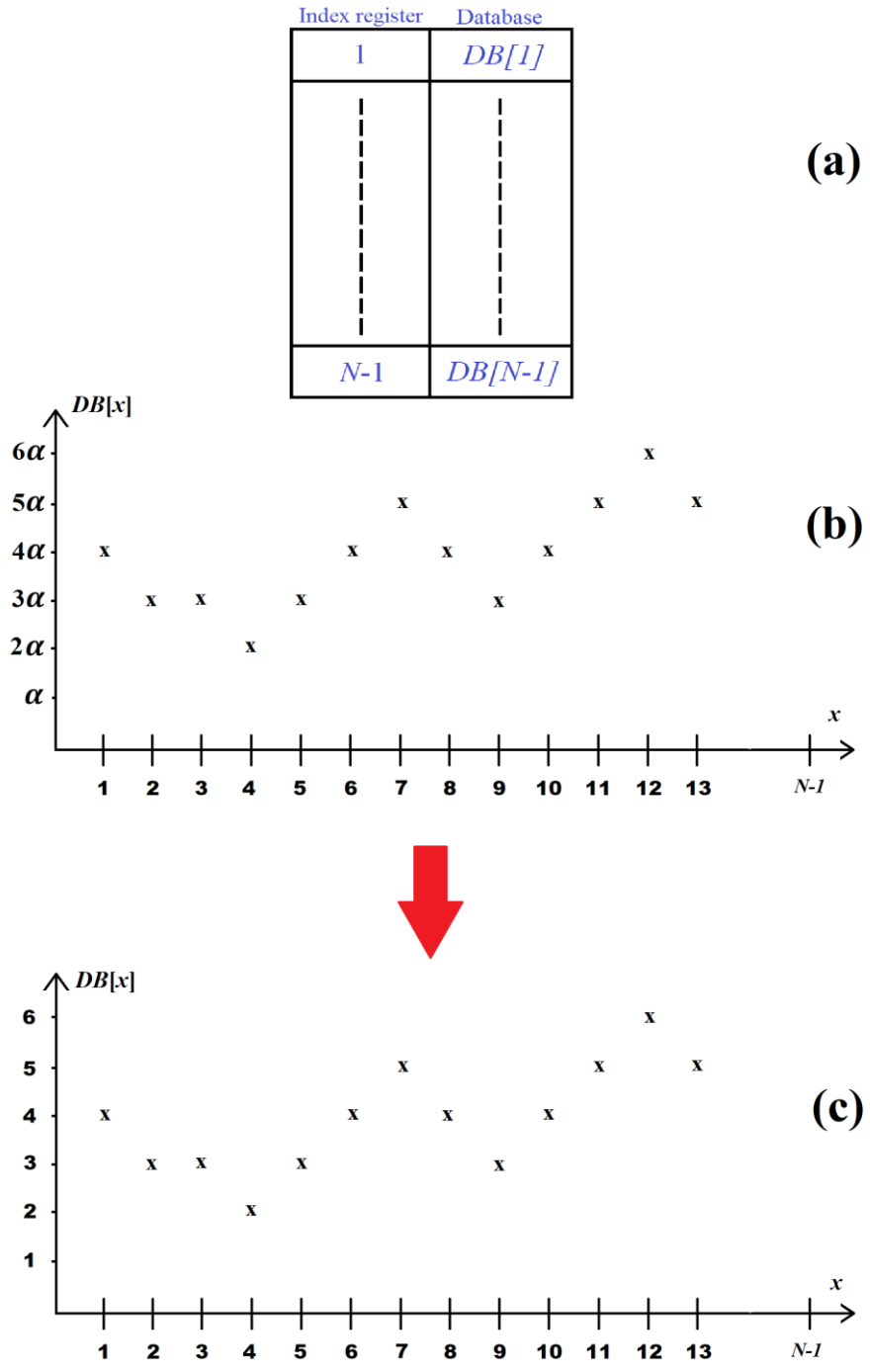


Figure 2.2: (a) The index registers and their database values. The conversion of the database values of the y-axis from the original unit scale form (b) to the new unit scale (c).

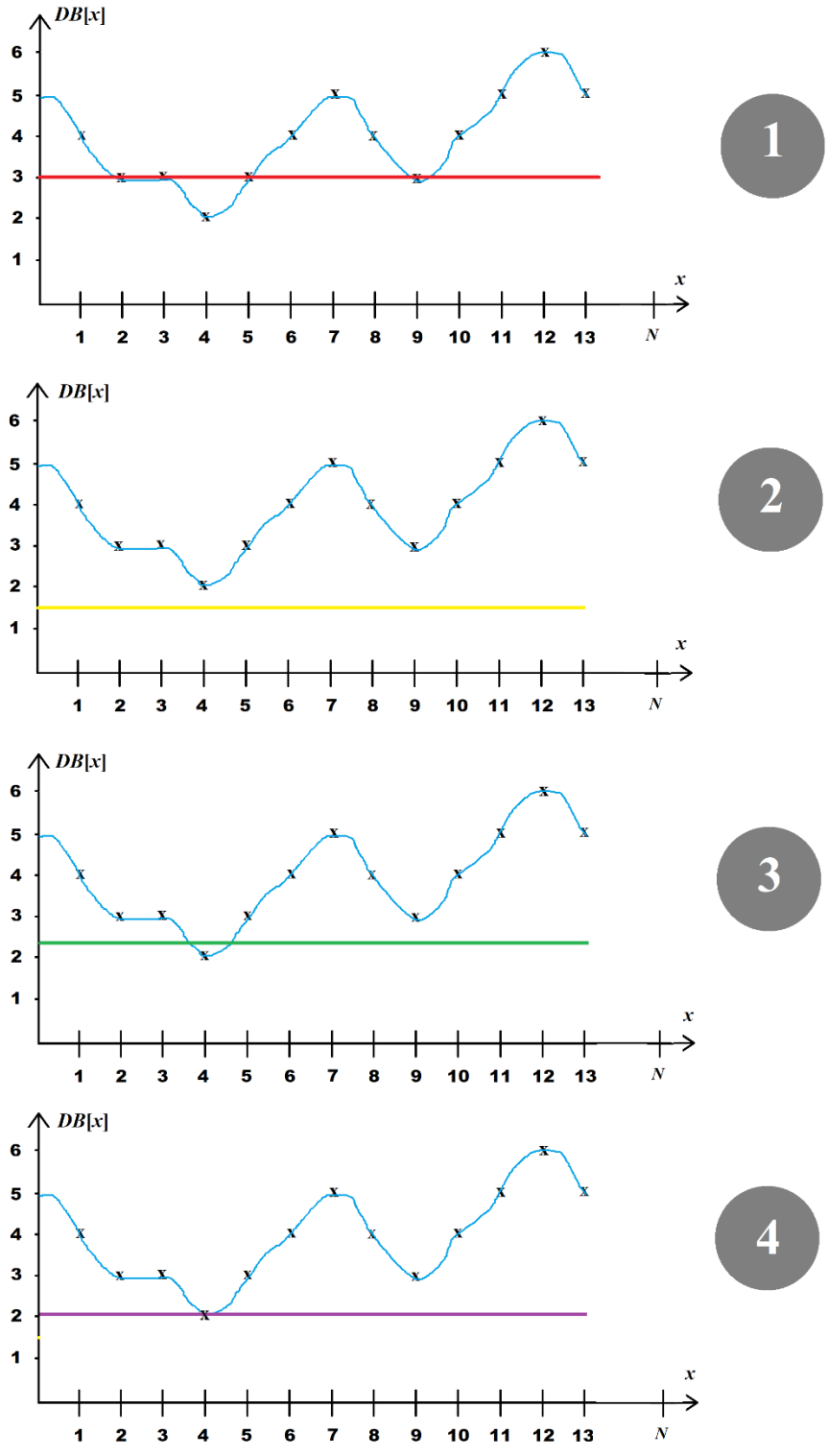


Figure 2.3: The description of the QEVSA as a quantum minimum searching algorithm.

2.3 Introducing the QOANCDS: the generalization of the quantum existence testing to quantum relation testing,

For many optimization problems, we cannot expect that the database will always have a CDS, which means that there exist certain cases where the database is not organized and fully random. This implies that there exist at least one index register x such that $|DB[x+1] - DB[x]|$ can be written as Eq. (2.8). Figure 2.4 illustrates an example of a Non-Continuous Database Structure (NCDS).

$$|DB[y+1] - DB[y]| = m \quad m \in \mathbb{N} - \{0,1\}. \tag{2.8}$$

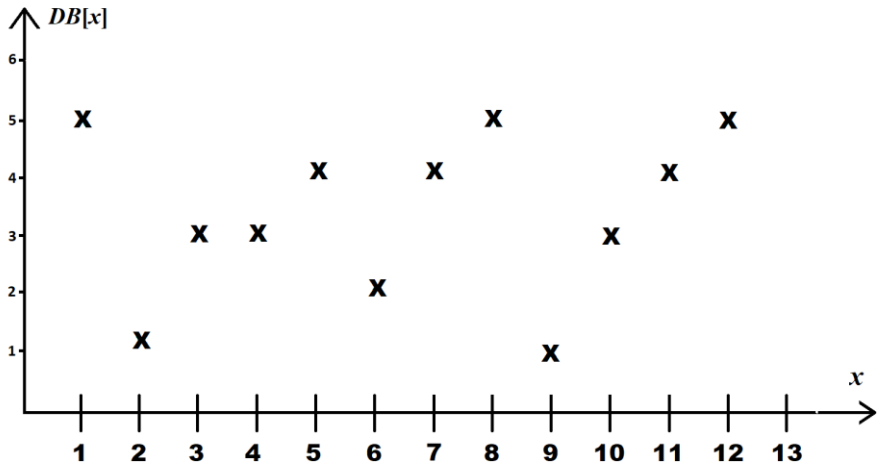


Figure 2.4: Scheme illustrating the structure of non-continuous database.

Now, let's assume that the database is NCDS and that we aim to optimize a constrained goal function. As an explicative example, see Figure 2.5, where the marked points with orange squares represent the database values that satisfy the proposed engineering constraint.

It is worth emphasizing that finding the optimum extreme solution in NCDS is a high-value problem where QEVSA cannot perform properly this type of search because the QET function embedded in the QEVSA does not support the search in the NCDS. For this sake, we present a new quantum method, quantum relation testing (QRT), which is an extended version of the QET. The QRT(*ref*, *R*) function returns only an answer (YES or NO) whether there are at least one or

more database entries in a given region of the database with respect to the applied index relation R and the reference value *ref*, where the parameter *ref* refers to the updated value which divides the database into two vertical sections, and the index relation R refers to the used relation, i.e. the index R may have the relation of greater ($>$) or greater or equal ($\geq$) which refers to finding the maximum optimum result, less ($<$) or less or equal ($\leq$) which refers to finding the minimum optimum result, or equality ($=$) which refers to applying the QET.

A detailed description of the various options for the relation R with respect to the given optimization problem target and the type of the database structure are summarized in Table 2.1.

Table 2.1: Different choices of the relation R according to the given optimization problem.

	Optimization Problem Target	**Type of Database Structure**
"="	Achieving the extreme optimum solution value	NCDS
$<$ and $\leq$	Achieving the minimum optimum solution value	NCDS
$>$ and $\geq$	Achieving the maximum optimum solution value	NCDS

It is crucial to stress that if we consider the index relation R is "=", this means that the QRT(*ref*, "=") operates similarly as QET(*ref*), such that the relation R has no effect on the database type, i.e. the database is being continuous if "=" is assigned to R. While employing one of the remaining relations (" $<$ ", " $\leq$ ", " $>$ ", or " $\geq$ ") implies that the type of the database is non-continuous.

The main question that is going to be answered are as follows,

- What does the QRT do?
- How the QRT exceeds the QET?
- Explain why the computational complexity of the QRT is similar to the computational complexity of QET?
- What are the similarities and the differences between the QET and the QRT?

The computational complexity of the QRT is strongly connected to the classical and quantum uncertainty of the application. Similar to the QET function, the QRT is merely a hidden QPE. It should be noticed that the QRT determines whether a certain item exists in a certain section of the database that gratify both $\Omega \neq 0$ ($M \neq 0$) and the relation R. Let $\tilde{\Omega}$ denotes the measured angle between the initial qregister and the horizontal axis $|\alpha>$, i.e. the parameter $\tilde{\Omega}$ indicates the measured angle.

The total number of qubits used for the physical implementation of the QRT is denoted by n_R, it can be described as follows.

$$n_R = c_R + p_R. \tag{2.9}$$

where, c_R and p_R refer respectively to the number of qubits needed for classical accuracy demand and quantum uncertainty of the QRT. It is substantial to point out that these two quantities are distinct and not mutually influential.

Let $P_{\epsilon R}$ be the error probability of QRT originated from quantum uncertainty of the phase estimation. It is interesting to note that there is a connection between the p_R and $P_{\epsilon R}$ which makes it easy to estimate the approximate value of p_R.

The expression of $P_{\epsilon R}$ is based on the following assumptions: if $M = 0$ then $\tilde{\Omega}$ is accurately measured, corresponding to the idealistic QPE. Inaccurate measurement happens when $\tilde{\Omega}$ is detected as zero, even though it is not equal to zero, it is worth mentioning that incorporating the relation R in the QPE does not affect the $P_{\epsilon R}$, one can express the $P_{\epsilon R}$ as

$$P_{\epsilon R} = \underbrace{P\left(\left(\tilde{\Omega} \neq 0\right)\middle|\Omega = 0\right)}_{\equiv 0} P(\Omega = 0) + \sum_{\Omega \neq 0} P\left(\tilde{\Omega} = 0|\Omega\right)P(\Omega). \tag{2.10}$$

Thus, one can easily verify that

$$P_{\epsilon R} = P_{\epsilon E}. \tag{2.11}$$

The remaining quantum certainty estimation of Eq. (2.10) for the QRT is similar to the QET represented in [40-41]. Consequently, one can conclude,

$$p_R = p_E = log_2(2\pi) + log_2\left(\frac{1}{8\widetilde{P}_\varepsilon}\right). \tag{2.12}$$

Additionally, it is substantial to emphasize that involving R in the QPE does not influence the classical accuracy demand of the application. Thus, the classical certainty of the QET and QRT are equal,

$$c_R = c_E = \frac{a}{2} - 1. \tag{2.13}$$

The computational complexity of QRT in this regard is equal to that of QET.

The QOANCDS finds the extreme (maximum or minimum) optimal value of an unconstrained objective function, the key to the effectiveness of the QOANCDS stems from merging the QRT with the BSA. It is interesting to note that the type of the relation R determines the type of the database structure (CDS or NCDS); The computing complexity of the QOANCDS is similar to the QEVSA.

The working methodology of the QOANCDS is given as follows: in case of searching the optimum minimum result of an unconstraint goal function where the database is NCDS, the QOANCDS splits the region ΔF into two sub-regions (the upper and lower region), then, selects the bellow region and apply the quantum relation testing if the QRT function finds at least one or more database entries which fulfill the relation R to reference ref, we apply another subdivision and the same process can be repeated until we find the optimum minimum solution, else we jump to the upper region and we continue using the aforementioned search process until we get the optimum minimum result or the algorithm stops.

2.4 Introducing the CQOA: the generalization of the quantum relation testing to constrained quantum relation testing

Now, let's assume that the database is a CDS and that we aim to optimize a constrained goal function. As an explicative example, see Figure 2.5, where the marked points with orange squares represent the database values that gratify the proposed engineering constraint. It is interesting to note that applying a constrained optimization search means eliminating all possible item candidates that do not satisfy the applied constraint from the database. This operation would transform the database structure from CDS to a NCDS. To that end, we present a new quantum algorithm which is an extended version of the QRT, called constrained quantum relation testing (CQRT). The $CQRT(ref, R, C)$ gives back only an answer whether there exist at least one or more database entries with respect to the applied relation R of the reference value *ref* and the constraint C in a certain region of the database.

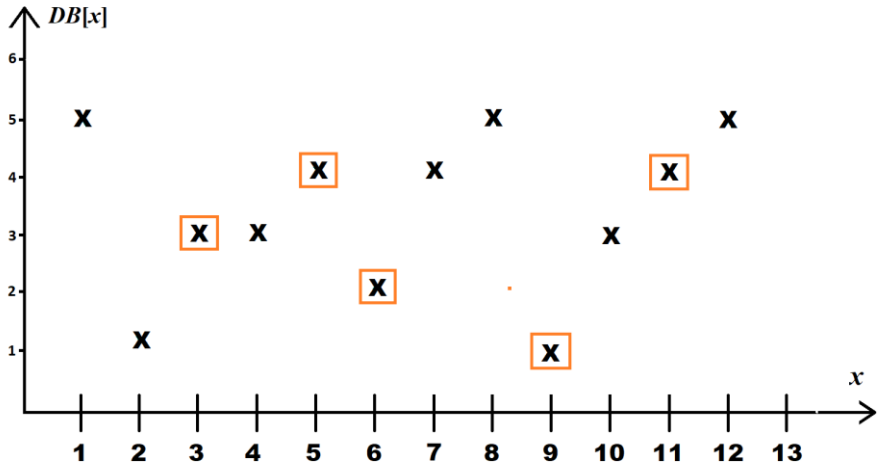


Figure 2.5: Scheme illustrating the structure of non-continuous database structure and the points where the constraint is fulfilled.

The CQRT's computational complexity is closely related to the application's classical and quantum accuracy. Similar way as before, the CQRT is nothing else than the QPE in disguise. The CQRT checks whether there exists a certain item in a certain region of the database which fulfills at the same time $\Omega \neq 0$ ($M \neq 0$) and $C = 1$, i.e. if the constraint C is fulfilled then $C = 1$, else $C = 0$. It is worth deriving a complete calculation for the optimum number of qubits n_{CR} needed for the CQRT. The number of qubits needed for classical accuracy is denoted by c_{CR} and the

number of qubits required for quantum certainty is defined by p_{CR}, these two quantities are independent and they do not influence each other. The n_{CR} can be written as

$$n_{CR} = c_{CR} + p_{CR}. \tag{2.14}$$

It is interesting to note that involving the constraint *C* in QRT does not modify the conversion of QPE, as a result, the same derivation can be applied for CQRT for computing the classical accuracy. Thus the classical accuracy of QRT [40] is similar to the CQRT, one writes

$$c_{CR} = c_R = \frac{a}{2} - 1. \tag{2.15}$$

It is important to mention that if $M = 0$ then $P(\tilde{\Omega} = 0) = 1$ is always measured with certainty, it corresponds to the idealistic QPE. Inaccurate measurement occurs if $\tilde{\Omega}$ is detected as zero even $\Omega \neq 0$, and as the constraint *C* does not influence the measurement of $\tilde{\Omega}$, i.e. the value of the constraint *C* is always detected with certainty either holds or not, also involving *R* in the QPE does not influence the calculation of the required qubits needed to ensure the quantum certainty of the phase estimation. On the other hand, to give back a correct measurement result, the p_{CR} must be estimated to suppress the error from the measurement. The probability of error for the CQRT is denoted by $P_{\epsilon CR}$. It is interesting to note that there is a strong relationship between the p_{CR} and $P_{\epsilon CR}$ such that if we can estimate the upper bound of $P_{\epsilon CR}$, we can easily calculate the p_{CR}. The $P_{\epsilon CR}$ can be expressed as

$$P_{\epsilon CR} = \underbrace{P\left(\left(\tilde{\Omega} \neq 0 \cap C\right) \middle| \Omega = 0\right)}_{\equiv 0} P(\Omega = 0) + \sum_{\Omega \neq 0} P\left(\left(\tilde{\Omega} = 0 \cap C = 1\right) \middle| \Omega\right) P(\Omega) \tag{2.16}$$

The measurement of the phase $\tilde{\Omega}$ does not influence the exact value of the constraint *C*. For this sake, for every $\Omega \neq 0$, $P\left(\left(\tilde{\Omega} = 0 \cap C = 1\right) \middle| \Omega\right) = P\left(\tilde{\Omega} = 0 | \Omega\right)$. Thus, one can rewrites Eq. (2.16) as

$$P_{\epsilon CR} = \underbrace{P\left(\left(\tilde{\Omega} \neq 0 \cap C\right) \middle| \Omega = 0\right)}_{\equiv 0} P(\Omega = 0) + \sum_{\Omega \neq 0} P\left(\tilde{\Omega} = 0 | \Omega\right) P(\Omega). \tag{2.17}$$

The remaining derivations of $P_{\epsilon CR}$ Eq. (2.17) are similar to $P_{\epsilon R}$ discussed in [40] because $P_{\epsilon CR} = P_{\epsilon R}$ (The remaining quantum certainty evaluation of formula Eq. (2.17) for the CQRT is

similar to the QRT). To this end, the number of qubits required for quantum certainty in both cases, the QRT and CQRT are equal, the resulting p_{CR} can be expressed as

$$p_{CR} = p_R = log_2(2\pi) + log_2\left(\frac{1}{8\breve{P}_\varepsilon}\right). \tag{2.18}$$

As we already mentioned, the relation *R* and the constraint *C* do not affect the classical and the quantum uncertainty of the CQRT function. In other words, from Eq. (2.15) and Eq. (2.18), the estimation of the computational complexity of the CQRT is therefore straightforwardly yielded. To this end, the computational complexity of the QET equals the computational complexity of CQRT.

The CQOA finds the optimum extreme value (minimum or maximum) of a constraint goal function. The CQOA has a stochastic behavior and combines the $CQRT(ref, R, C)$ and the BSA. The CQRT allows adapting the BSA so that it is suitable for an unsorted database, as well as continuous and non-continuous database structures. From a practical point of view. The CQOA is given as follows,

---

1. We start with $S = 0$: $T_{min1} = T_{min0}$, $T_{max1} = T_{max0}$, and $\Delta T = T_{max0} - T_{min0}$
2. $S = S + 1$
3. $T_{med\,S} = T_{minS} + \left[\frac{T_{maxS} - T_{minS}}{2}\right]$
4. $flag = QET\left(CQRT\left(T_{med\,S}, R, C\right)\right)$:
   - $flag = Yes$, then $T_{max\,S+1} = T_{med\,S}$, $T_{\min S+1} = T_{\min S}$
   - Else $T_{max\,S+1} = T_{max\,S}$, $T_{min\,S+1} = T_{med\,S}$
5. If $S < log_2(T)$, then go to 2, else stop and $y_{opt} = T_{med\,S}$

Figure 2.6 and Figure 2.7 present the working mechanism of the CQOA in the case of searching for the optimum minimum solution of a constraint goal function.

In a more general case, for multiple constraint values, C_1, C_2,..., C_n, the constraint quantum relation testing became ($CQRT(R, ref, C_1, C_2,..., C_n)$). It is worth mentioning that applying the CQRT with taking into consideration the constraints C_1, C_2,..., C_n, means practically filtering several items in the database, i.e. eliminating possible item candidates from the database.

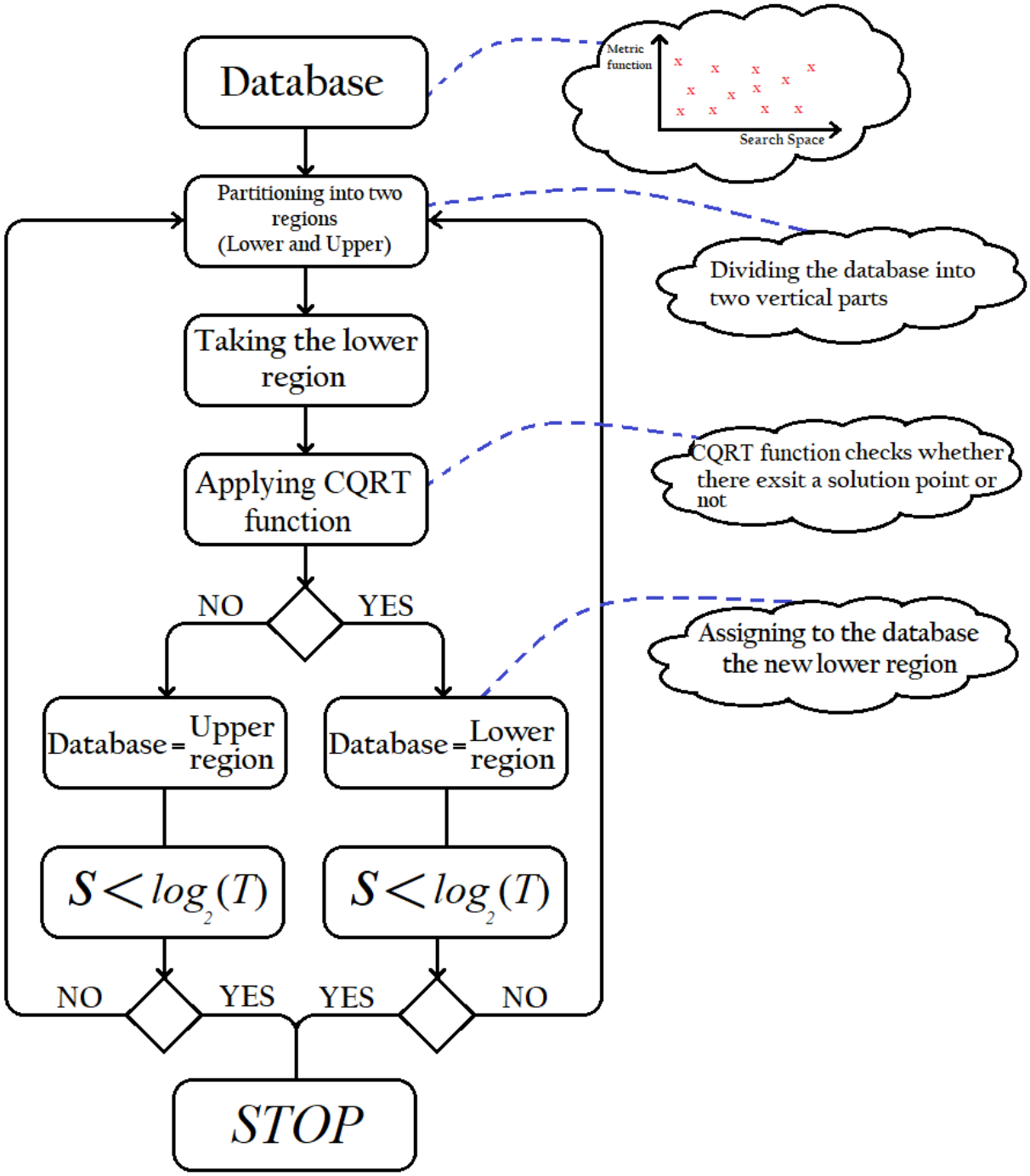


Figure 2.6: The working mechanism of the CQOA in case of searching the optimum minimum result of a constraint goal function.

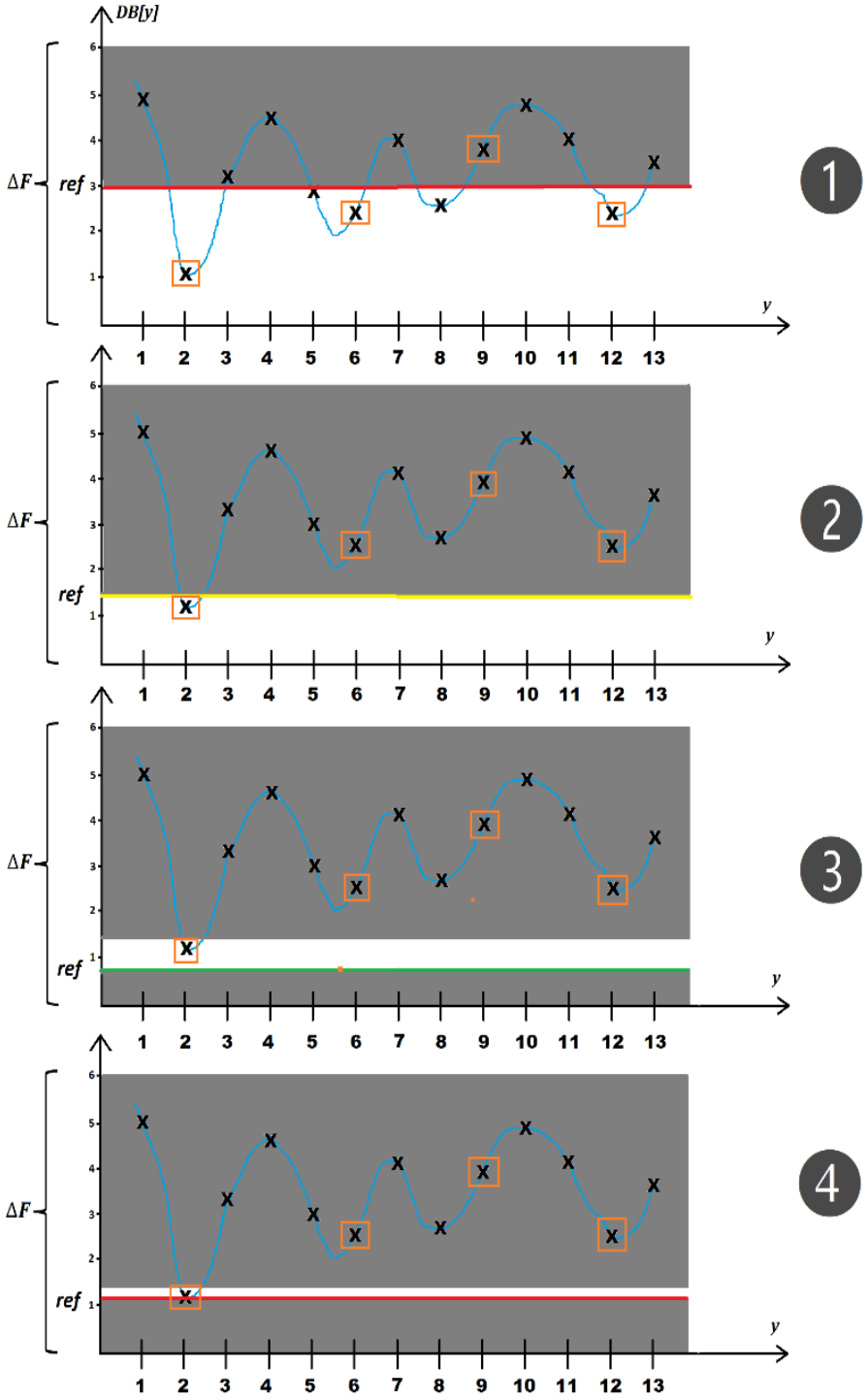


Figure 2.7: The working mechanism of the CQOA in case of searching the optimum minimum result of a constraint goal function (orange circles represent the values that satisfy the constraint.)

2.5 Comparison of the quantum optimization algorithms

Based on what has been discussed, it is straightforward to notice that the special quantum core functions used in each quantum optimization algorithm (QEVSA, QOANCDS, and CQOA) are connected by $QET \subseteq QRT \subseteq CQRT$. Figure 2.8 presents the relationship between the already presented quantum optimization methods.

The QET operates in the CDS while the QRT and CQRT may handle the search in a CDS and NCDS. Table 2.2 summarizes the similarities and differences between the QEVSA, QOANCDS, and CQOA.

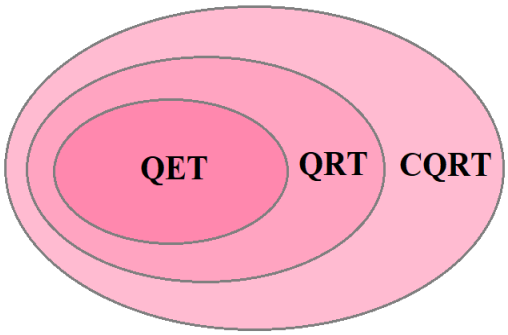


Figure 2.8: The relationship between the QET, QRT, and CQRT.

Table 2.2: The similarities and differences between the QEVSA, QOANCD, and CQOA.

	QEVSA	**QOANCDS**	**CQOA**
Type of objective function (OF)	Unconstraint OF	Unconstraint OF	Constraint OF
Database Structure (sorted/unsorted)	Unsorted database	Unsorted database	Unsorted database
Computational complexity	$O\left(log_2(T)log_2{}^3(\sqrt{N})\right)$	$O\left(log_2(T)log_2{}^3(\sqrt{N})\right)$	$O\left(log_2(T)log_2{}^3(\sqrt{N})\right)$
The BSA is combined with	QET	QRT	CQRT
Database type (CDS/NCDS)	CDS	CDS/NCDS	CDS/NCDS

It is very interesting to point out that the quantum algorithms exploit the notion of quantum superposition which constitutes one of the pillars for guarantying high speed up and accuracy, as well as very low computational complexity. For this sake, the convergence of each algorithm does not depends on the amount of data to be processed. It is also important to note that the parameter T is the size of the range within the logarithm search runs. It depends on two parameters, the step size α as of the BSA, it is the minimum distance between any two values of the goal functions T_i and T_j corresponding to two different indexes registers i and j presented by Eq (2.19), and the region of the values of the goal function ΔT. The value of T is calculated as follows,

$$\alpha = \min_{\forall i,j} \left| (T_i - T_j) \right|, \tag{2.19}$$

$$T = \frac{T_{max} - T_{min}}{\alpha}. \tag{2.20}$$

On the other hand, the configuration of the stochastic parameter T for the QEVSA, QOANCDS, and CQOA can be performed using Eq. (2.19) and Eq. (2.20).

2.6 Computational complexity comparison of optimization algorithms

In this subsection, we compare the QEVSA with some classical optimization algorithms. We present three well-known algorithms: the randomized, the exhaustive, and the sequential-searching algorithms which are generally viewed as references and cornerstones with regards to finding solutions [43]. Firstly, the randomized approach anticipates the solution guided by a given knowledge and it is not perceived as an optimal solution because it randomly searches for one solution in one step from the space in which the duration of time may be less reasonable [44], on the other hand, the exhaustive algorithm examines every possible solution and leads to an optimal result, it checks all the $O(N)$ steps which is time-consuming until it yields an accurate result. A large number of iterations are required when compared to the number of queries, hence, its time-consuming nature**.** Furthermore, the sequential searching method exhibits a similar degree of computational complexity to the randomized method, using $O(const)$ or $O(1)$.

By comparing the computational complexity of the quantum, randomized, exhaustive, and sequential methods, the quantum solution requires only $O\left(log_2(T) {log_2}^3\left(\sqrt{N}\right)\right)$ steps. On the

other hand, the exhaustive solution needs $O(N)$. With regard to the randomized solution, it uses fewer steps so is faster than the three former solutions, which means that the computational complexity of the random method is less than for the other ones. Moreover, it does not determine the optimum solution, for the computational complexity of the sequential method is similar to the randomized algorithm. In contrast, the exhaustive and quantum solutions require more computation steps, but the quantum solution is always preferred because it requires significantly less computation than the exhaustive method. Table 2.3 presents different classical optimization algorithms and compares them with the QEVSA in terms of computational complexity and accuracy. In the light of what has been shown, in Figure 2.9, the classical strategy needs more computational computing, while the quantum solution requires significantly fewer computations, which reduces the complexity and the time for finding the optimum result.

Table 2.3: Comparison between the classical heuristic algorithms and the QEVSA

Methods	Accuracy	Computational complexity
Randomized	Very rough	$O(const)$ or $O(1)$
Sequential	Very rough	$O(const)$ or $O(1)$
Exhaustive	Exact	$O(N)$
QEVSA	Close exact	$O\left(log_2(T) {log_2}^3(\sqrt{N})\right)$

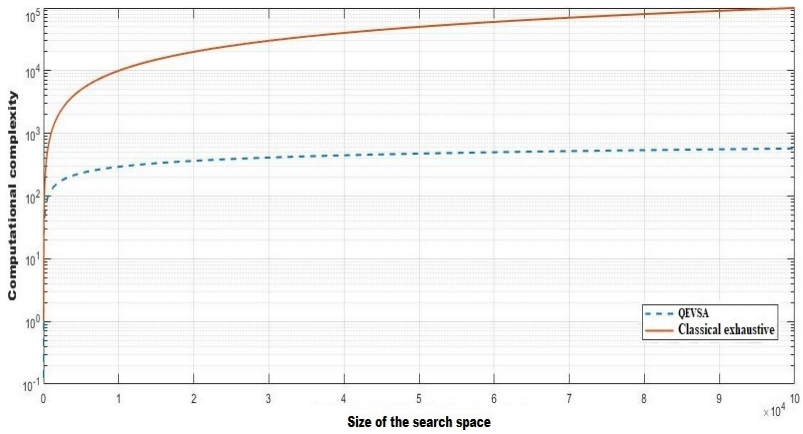


Figure 2.9: Comparison between the computational complexity of the classical optimization algorithm and the proposed QEVSA.

2.7 Conclusion

An extended version of the Quantum Extreme Value Searching Algorithm (QEVSA) called Quantum Optimization Algorithm for Non-Continuous Database Structure (QOANCDS) has been developed that finds the extreme solution of an unconstrained objective function or an unsorted database. The classical and quantum certainty of the QOANCDS, as well as its computational complexity, have been investigated. Next, we showed how the functionalities of the QOANCDS exceed the QEVSA.

A new improved version of the QOANCDS called Constrained Quantum Optimization Algorithm (CQOA) has been introduced that finds the extreme solution of a constrained objective function or an unsorted database with respect to a certain constraint. The classical and quantum certainty of the CQOA, as well as its computational complexity, have been investigated. After, we showed how the functionalities of the QOANCDS exceed the QEVSA.

The relationship between the core quantum functions of the QEVSA, QOANCDS, and CQOA has been highlighted. An analytical study showed that the computational complexity of QEVSA, QOANCDS, and CQOA are equal $O\left(log_2(T){log_2}^3\left(\sqrt{N}\right)\right)$ and the QEVSA achieves a high computational complexity reduction compared with the existing classical optimization methods.

3 Quantum optimization of resource distribution management for multi-tasks

Suppose all classical machines that are working on classical laws of classical physics will disappear from our world and be replaced by quantum computers, so there will be an intensive need for developing new adaptive models. From a resource distribution management point of view, if the real-time decision-maker will be replaced by a quantum method, so, the way becomes open to think how to implement a resource distribution management model with the new device since new hardware technique requires a new resource distribution process modeling.

3.1 Introduction

Chapter 3 applies the QEVSA in the RDM model, i.e. the multi-tasks case is considered, while Chapter 4 implements the QEVSA for an extended version of the RDM system, i.e. the multi-tasks and multi-subtasks case is considered. Chapter 5 incorporates the CQOA in the most general RDM model, i.e. multi-tasks and multi-subtasks case. While Chapter 6 exploits the CQOA in the RDM by considering queueing scenarios.

The rest of this Chapter is organized as follows. Section 3.2 describes the proposed RDM model for multi-tasks. Section 3.3 implements the QEVSA in the suggested RDM system, Section 3.4 presents the configuration of the QEVSA. Section 3.5 discusses the computational complexity of the applied quantum strategy. Section 3.6 concludes the paper with simulation results, and Section 3.7 is a conclusion of the chapter.

3.2 Description of the RDM system for multi-tasks

To model the general RDM system, we divided its functionalities into three main blocks as depicted in Figure 3.1:

Multiple task generator: Let G denote the total number of task type generators (i.e. we consider a multi-task scenario), each task generator has its own intensity distribution (e.g. exponential

distribution, etc) and produces identical tasks. A task is composed of a specified number of identical memory pieces called memory units.

Computing units: Let K be the total number of computing units used to serve the tasks, the computing units may have different theoretical capacities, such as the k^{th} computing unit has s_k as a theoretical capacity, let the number of running tasks from task type l on-the k^{th} computing unit is denoted by N_{kl}.

The processing rate of the k^{th} computing unit is denoted by β_k and computed as follows,

$$\beta_k = \frac{number\ of\ memory\ units}{second} \tag{3.1}$$

On the other hand, the processing delay of the actual load on the k^{th} computing unit can be calculated as,

$$\tau_k^{act} = \frac{\sum_{l=1}^{G} N_{kl}\, m_l}{\beta_k} \tag{3.2}$$

It is interesting to note that the tasks are processed sequentially inside the computing unit. Figure 3.2 illustrates the processing mechanism of tasks inside the computing unit.

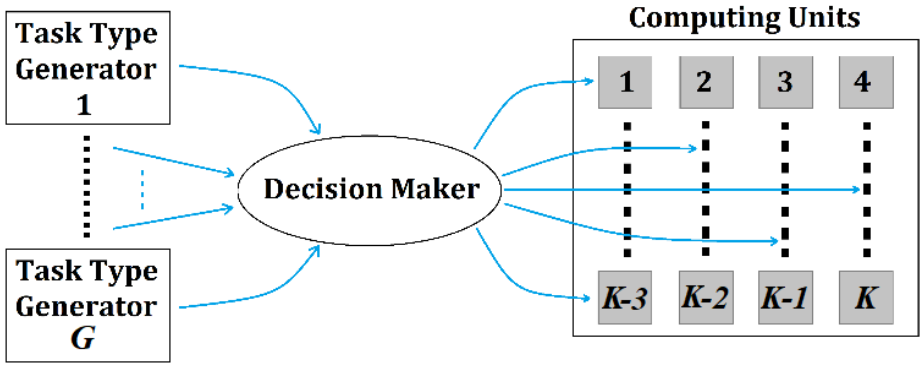


Figure 3.1: Resource distribution management architecture of multiple tasks.

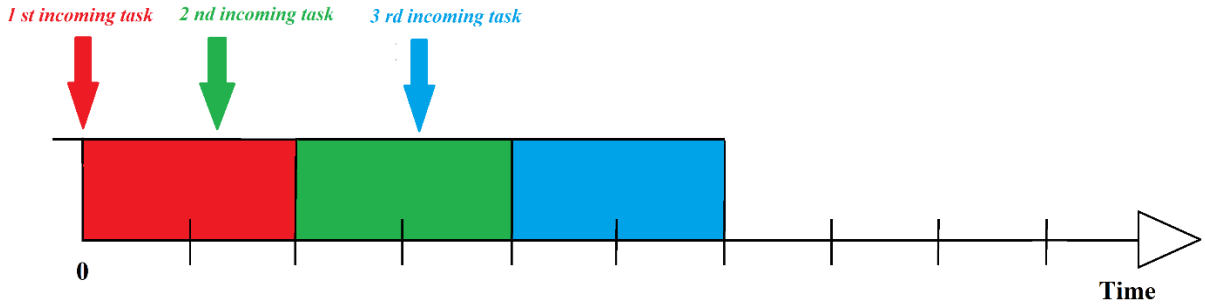


Figure 3.2: Processing mechanism of tasks inside a certain computing unit.

Decision maker: This block answers the question of whether to reject or to deploy a newly arrived task among the computing units to optimize the operation of the system? Optimization is achieved along with a certain goal function, i.e. certain metric.

3.3 Implementation of the QEVSA in the RDM system for multi-tasks

This subsection introduces how the QEVSA can be used as a computational infrastructure for the RDM system to reach less computational complexity and a high degree of accuracy as well as load balancing. This study does not discuss the resource requirement scheme implementation because the QET is a special form of QCA and the quantum gate circuit structure is well known, the modification will be only in the QEVSA.

Conserving the load balance among the computing units is an important property for many applications. For this sake, we have chosen to distribute the tasks uniformly among the computing units. Therefore, the variance of the relative load of the RDM system is used as a measurement metric. In the case of an optimal task distribution, if σ^2 tends to zero, then the tasks are distributed uniformly, otherwise, they are not. Therefore we apply the QEVSA as a QMSA, which allows finding the optimum deployment scenario that results in the minimum variance of the relative load. Our new approach handles the database as a function, i.e. the variance of the relative load. In this case, the goal function σ^2 replaces T. The proposed algorithm is now rewritten in detail:

---

1. We start with $S = 0 : {\sigma^2}_{min1} = {\sigma^2}_{min0}$, ${\sigma^2}_{max1} = {\sigma^2}_{max0}$, and $\Delta\sigma^2 = {\sigma^2}_{max0} - {\sigma^2}_{min0}$
2. $S = S + 1$

3. $\sigma^2{}_{med\,S} = \sigma^2{}_{minS} + \left[\frac{\sigma^2{}_{maxS} - \sigma^2{}_{minS}}{2}\right]$

4. $flag = QET\ (\sigma^2{}_{med\,S})$:

   - $flag = Yes,$ then $\sigma^2{}_{max\,S+1} = \sigma^2{}_{med\,S}$, $\sigma^2{}_{\min S+1} = \sigma^2{}_{\min S}$
   - Else $\sigma^2{}_{max\,S+1} = \sigma^2{}_{max\,S}$, $\sigma^2{}_{min\,S+1} = \sigma^2{}_{med\,S}$

5. If $S < log_2\,(T)$, then go to 2, else stop and $y_{opt} = \sigma^2{}_{med\,S}$

3.4 Configuration of the QEVSA

Applying the QMSA in finding the best scenario for every new incoming task requires computing the maximum number of steps needed to run the BSA, i.e. the parameter T.

It is worth noting that integrating the RDM model in the framework of a quantum system-level is not straightforward and may need a careful configuration of its parameters.

The maximum number of steps T needed to run the BSA embedded in the QMSA depends on two parameters which are the step size of the search space that is according to the proposed RDM model is the absolute value of the minimum nonzero difference between variances of any two scenarios α as presented in Eq. (3.3), where σ_i^2 replaces T_i in Eq. (2.19) and the size of the region of variance's values $\Delta\sigma^2 = \sigma^2{}_{max} - \sigma^2{}_{min}$, the expression of T is illustrated in Eq. (3.4),

$$\alpha = \min_{\forall\, i,j} \left|(\sigma_i^2 - \sigma_j^2)\right|, \tag{3.3}$$

$$T = \frac{\sigma^2{}_{max} - \sigma^2{}_{min}}{\alpha}. \tag{3.4}$$

- α - The minimum distance between variances of two scenarios depends on many parameters such as the task memory requirement, processing time, the workload, the arrival time distribution of the task, the number of the computing units, and their theoretical capacities. An illustrative example of α is presented in Figure 3.3.
- $\Delta\sigma^2$ - The possible region of the variance is bounded by 0 and 1.

The configuration of T is formulated as finding the minimum distance between variances of two scenarios according to Eq. (3.3). To that end, our key focus is to set up the α.

Let x_k be the total load of the k^{th} computing unit

$$x_k = \sum_{l=1}^{G} N_{kl}\, m_l. \tag{3.5}$$

Assume b_k be the relative load of the k^{th} computing unit,

$$b_k = \frac{\sum_{l=1}^{G} N_{kl}\, m_l}{s_k}. \tag{3.6}$$

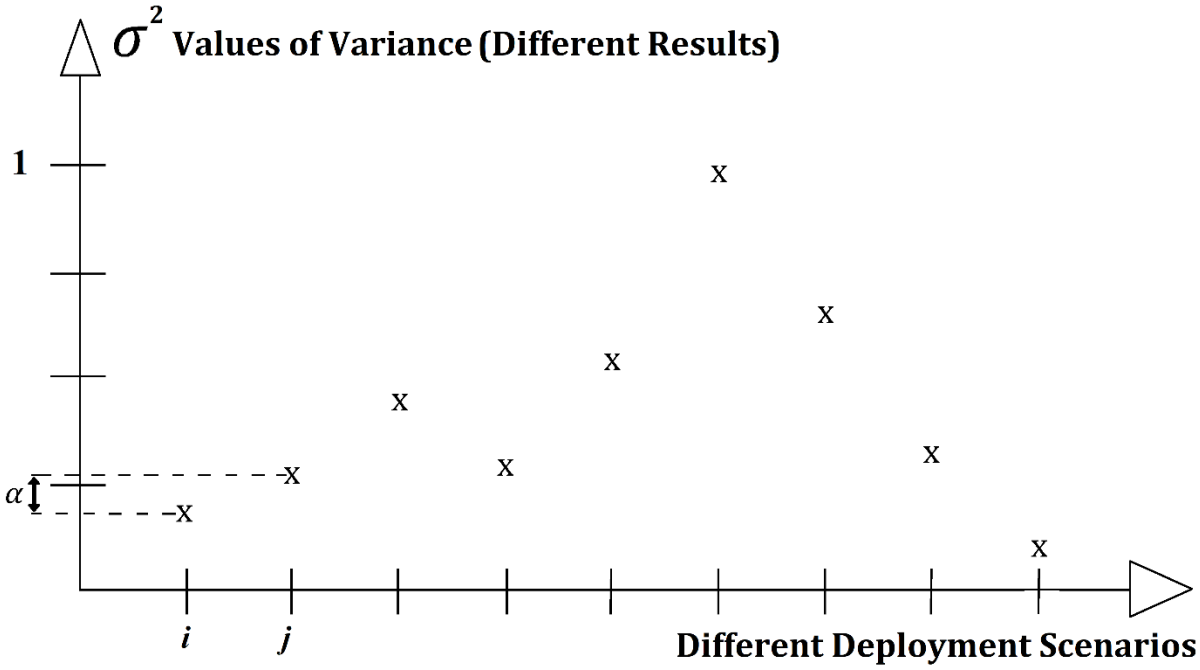


Figure 3.3: The horizontal axis presents all the possible deployment scenarios, while the vertical axis presents the borders of the variance function (different results), each possible scenario corresponds to a variance value. Computing the value of α requires selecting the minimum distance between variances of two deployment scenarios i and j.

Let $\bar{b}$ refer to the average relative load of the k^{th} computing unit,

$$\bar{b} = \frac{1}{K} \sum_{k=1}^{K} b_k\,. \tag{3.7}$$

Let s denote the class of the new task, the memory requirement of the new task is expressed by m_s while the i^{th} computing unit indicates the destination of the new incoming task. It is interesting to note that selecting the suitable computing unit for the new incoming task requires updating the value of the i^{th} computing unit, which implies that the relative load of the i^{th} computing unit $b_{\substack{k=i \\ l=s}}^{i,s}$ and the relative load of the remaining computing units $b_{k\neq i}$ can be expressed respectively as

$$b_{\substack{k=i \\ l=s}}^{i,s} = b_i + \frac{m_s}{s_i}, \tag{3.8}$$

$$\forall k \in \{1, \dots, K\} \backslash \{i\} \qquad b_{k\neq i} = b_k. \tag{3.9}$$

The average of the relative load of the system when the i^{th} computing unit receives the new incoming task from class s is given as follows,

$$\overline{b_{l=s}} = \frac{1}{K}\sum_{k\neq i}^{K} b_k^i + \frac{1}{K}\, b_{\substack{k=i \\ l=s}}^{i,s} \ . \tag{3.10}$$

One can verify that

$$\overline{b_{l=s}} = \bar{b} + \frac{1}{K}\,\frac{m_s}{s_i} \ . \tag{3.11}$$

One can write the variance of the relative load of the system when the i^{th} computing unit receives the new incoming task s as

$$\begin{aligned} \sigma_i^2 &= \frac{1}{K}\sum_{k=1}^{K}\left(\overline{b_{l=s}} - b_k^i\right)^2 \\ &= \frac{1}{K}\sum_{k\neq i}^{K}\left(\overline{b_{l=s}} - b_k\right)^2 + \frac{1}{K}\left[\overline{b_{l=s}} - \left[b_i + \frac{m_s}{s_i}\right]\right]^2. \end{aligned} \tag{3.12}$$

It is reasonable to assume that the number of computing units K is very large, one obtains

$$
\begin{aligned}
\sigma_i^2 &= \frac{1}{K}\sum_{k\neq i}^{K}\left(\frac{1}{K}\,\frac{m_s}{s_i} - b_k\right)^2 + \frac{1}{K}\left[\frac{1}{K}\,\frac{m_s}{s_i} - \left[b_i + \frac{m_s}{s_i}\right]\right]^2 \\
&= \frac{1}{K}\sum_{k\neq i}^{K}(-b_k)^2 + \frac{1}{K}\left[\left(\frac{1}{K} - 1\right)\frac{m_s}{s_i} - b_i\right]^2 \\
&= \frac{1}{K}\sum_{k\neq i}^{K} {b_k}^2 + \frac{1}{K}\left(\frac{m_s}{s_i}\right)^2 + \frac{1}{K}2b_i\left(\frac{m_s}{s_i}\right) + \frac{1}{K}\,{b_i}^2 \\
&= \frac{1}{K}\sum_{k=1}^{K} {b_k}^2 + \frac{1}{K}\left(\left(\frac{m_s}{s_i}\right)^2 + 2b_i\left(\frac{m_s}{s_i}\right)\right).
\end{aligned}
\tag{3.13}
$$

Finally, The difference between two dissimilar relative load variances (which is needed to calculate α) can be formulated as

$$
\sigma_i^2 - \sigma_j^2 = \frac{1}{K}\left[\left(\frac{m_s}{s_i}\right)^2 - \left(\frac{m_s}{s_j}\right)^2 + 2m_s\left(\frac{\sum_{l=1}^{G} N_{il}m_l}{s_i} - \frac{\sum_{l=1}^{G} N_{jl}m_l}{s_j}\right)\right]. \tag{3.14}
$$

To compute the local minimum α, one can verify that $\left|(\sigma_i^2 - \sigma_j^2)\right|$ has to be evaluated at most $K(K-1)/2$ times. This means that for $K > 3$, calculating α requires more evaluations to find the minimum of σ_i^2. Here we propose another alternative. Instead, we determine the global minimum of α over all the possible configurations in advance, before the RDM starts operation and this global α is used at each decision.

Obviously, this is an engineering tradeoff. On one hand, the global α will be the lover bound of the actual α. Therefore, the BSA will require more steps (but due to the nature of the log function this will not cause a real increase in computation complexity). On the other hand, we do not waste time during the real-time decisions by computing the local α!

As it is shown in Eq. (3.14), the value of $\sigma_i^2 - \sigma_j^2$ is mainly affected by the relative load of the i^{th} and j^{th} computing units, and more precisely with the values of the variables $N_{i1}, N_{i2}, \ldots, N_{iG}$, and $N_{j1}, N_{j2}, \ldots, N_{jG}$. Unfortunately, we cannot investigate the nonzero minimum value of $\left|(\sigma_i^2 - \sigma_j^2)\right|$ because the variables $N_{i1}, N_{i2}, \ldots, N_{iG}, N_{j1}, N_{j2}, \ldots, N_{jG}$ have a linear combination in

2G dimensional space. For this reason, we replace the absolute value function in Eq. (3.3) with the square of the expression of $\sigma_i^2 - \sigma_j^2$, as it is obtained as follows,

$$f(i,j) = \left(\sigma_i^2 - \sigma_j^2\right)^2 = \left(\frac{1}{K}\left[\left(\frac{m_s}{s_i}\right)^2 - \left(\frac{m_s}{s_j}\right)^2 + 2m_s\left(\frac{\sum_{l=1}^{G} N_{il}m_l}{{s_i}^2} - \frac{\sum_{l=1}^{G} N_{jl}m_l}{{s_j}^2}\right)\right]\right)^2. \quad (3.15)$$

To find the smallest non-zero difference between variances of two different assignments, it is enough to investigate $f(i,j) = 0$. It is important to mention that for any distribution, the difference between variances of two different scenarios depends only on the i^{th} and j^{th} computing units, we note that the value of α is not trivial, i.e. non-zero and $\sigma_i^2 \neq \sigma_j^2$. On the other hand, it is noticeable that $\left(\sigma_i^2 - \sigma_j^2\right)^2 \geq 0$. Therefore it is enough to investigate $\left(\sigma_i^2 - \sigma_j^2\right)^2 = 0$, this expression derives an important property of the minimum points (the variables N_{i1}, N_{i2},…, N_{iG}, and N_{j1}, N_{j2},…, N_{jG} are linearly dependent), i.e. the minimum places of the function $f = \left(\sigma_i^2 - \sigma_j^2\right)^2$ are situated in a hyperplane.

In the general case, to determine the desired α, it is needed to fulfill a number of restrictions determined by choosing the suitable range of N_{i1}, N_{i2},…, N_{iG}, and N_{j1}, N_{j2},…, N_{jG}. The function *f* is continuous, however, from the RDM point of view N_{i1}, N_{i2},…, N_{iG}, and N_{j1}, N_{j2},…, N_{jG} must be integer numbers. The first step is to assign integer numbers different than zero to all the variables except N_{il} (or N_{jl}), then compute the value of N_{il} (or N_{jl}), here, at this stage, there are two options whether the N_{il} (or N_{jl}) value is an integer or not.

- **1st case: if N_{il} (or N_{jl}) is an integer**

The minimum distance between the variances of two scenarios α equals zero and the values of N_{i1}, N_{i2},…, N_{iG}, and N_{j1}, N_{j2},…, N_{jG} are not appropriate solutions for α, so, in this case, the value of N_{il} (or N_{jl}) can be modified, by increasing or decreasing one of them, i.e. assigning the value of $N_{il} + 1$ or $N_{il} - 1$ to N_{il}, (or assigning the value of $N_{jl} + 1$ or $N_{jl} - 1$ to N_{jl}). The most important thing is to choose only one assignment which results in the minimum value of α, let us investigate the value of f when we assign the value $N_{il} + 1$ or $N_{il} - 1$ to N_{il}, (or assigning the

value of $N_{jl}+1$ or $N_{jl}-1$ to N_{jl}). Using the result stated in Eq. 3.12, let's prove that for a given task $l=r$, $f_{N_{ir+1}}=f_{N_{ir-1}}$.

In the case N_{il} is an integer where $l=r$, and considering the claim $\left(\sigma_i^2-\sigma_j^2\right)^2=0$. The expression of $f_{N_{ir+1}}$ is given as follows,

$$
\begin{aligned}
f_{N_{ir+1}} &= \left(\frac{1}{K}\left[\left(\frac{m_s}{s_i}\right)^2-\left(\frac{m_s}{s_j}\right)^2+2m_s\left(\frac{\sum_{l\neq r}^{G}N_{il}m_l}{{s_i}^2}+\frac{(N_{ir}+1)m_r}{{s_i}^2}-\frac{\sum_{l=1}^{G}N_{jl}m_l}{{s_j}^2}\right)\right]\right)^2 \\
f_{N_{ir+1}} &= \left(\frac{1}{K}\left[\left(\frac{m_s}{s_i}\right)^2-\left(\frac{m_s}{s_j}\right)^2+2m_s\left(\frac{\sum_{l=1}^{G}N_{il}m_l}{{s_i}^2}+\frac{m_r}{{s_i}^2}-\frac{\sum_{l=1}^{G}N_{jl}m_l}{{s_j}^2}\right)\right]\right)^2 \\
&= \left(\frac{1}{K}\frac{2m_sm_r}{{s_i}^2}\right)^2.
\end{aligned}
\tag{3.16}
$$

On the other hand, one can verify that $f_{N_{ir-1}}$ can be expressed as

$$
\begin{aligned}
f_{N_{ir-1}} &= \left(\frac{1}{K}\left[\left(\frac{m_s}{s_i}\right)^2-\left(\frac{m_s}{s_j}\right)^2+2m_s\left(\frac{\sum_{l\neq r}^{G}N_{il}m_l}{{s_i}^2}+\frac{(N_{ir}-1)m_r}{{s_i}^2}-\frac{\sum_{l=1}^{G}N_{jl}m_l}{{s_j}^2}\right)\right]\right)^2 \\
f_{N_{ir-1}} &= \left(\frac{1}{K}\left[\left(\frac{m_s}{s_i}\right)^2-\left(\frac{m_s}{s_j}\right)^2+2m_s\left(\frac{\sum_{l=1}^{G}N_{il}m_l}{{s_i}^2}-\frac{m_r}{{s_i}^2}-\frac{\sum_{l=1}^{G}N_{jl}m_l}{{s_j}^2}\right)\right]\right)^2 \\
&= \left(-\frac{1}{K}\frac{2m_sm_r}{{s_i}^2}\right)^2 \\
&= \left(\frac{1}{K}\frac{2m_sm_r}{{s_i}^2}\right)^2.
\end{aligned}
\tag{3.17}
$$

Based on the result of Eq. (3.16) and Eq. (3.17), we conclude that,

$$f_{N_{ir+1}}=f_{N_{ir-1}}. \tag{3.18}$$

Eq. (3.18) states that the cross-section of the hyperplane is symmetric on the minimum places of $\left(\sigma_i^2-\sigma_j^2\right)^2$ in *2G* dimensional space.

- **2nd case: if N_{il} (or N_{jl}) is not an integer**

Since f is a monotonous function, so, the nearest integer can be assigned to N_{il} (or N_{jl}), in this case, the parameter α does not equal zero.

To illustrate the location of the solutions of the $\left(\sigma_i^2 - \sigma_j^2\right)^2 = 0$, in case of handling only one task type generator. A simulation example is built to determine the location of the minimum places of α. Let's consider the following parameters, one task type generator is assumed, the memory requirement of the task is denoted by m, in this case, the Eq. (3.15) will be rewritten as Eq. (3.19), a detailed demonstration is presented in Appendix,

$$min\left(\sigma_i^2 - \sigma_j^2\right)^2 \cong min\left(\frac{1}{K}\left[\left(\frac{m}{s_i}\right)^2 (1 + 2N_i) - \left(\frac{m}{s_j}\right)^2 (1 + 2N_j)\right]\right)^2. \quad (3.19)$$

where N_i and N_j are the number of tasks respectively in the i^{th} unit and the j^{th} unit. For the simulation environment, the following setup is taken into consideration, $m = 1$, ${s_i}^2 = 10$ and ${s_j}^2 = 10$, the range of N_i and N_j is between -30 and 30. As reported in Figure 3.4 and Figure 3.5 the solution of the minimum distance between two different scenarios is situated in a line, which means that all the appropriate solutions are situated in the middle of the dark blue region, the area within the red border.

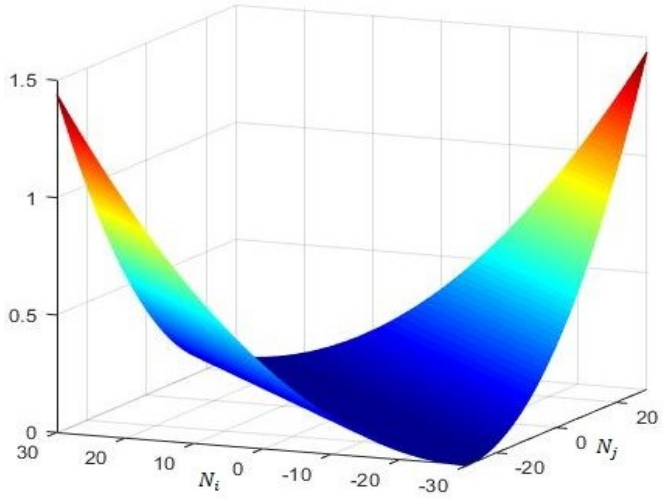


Figure 3.4: The shape of the function f with respect to N_i and N_j, in case, setting up the following parameters $m = 1$, ${s_i}^2 = 10$, ${s_j}^2 = 10$.

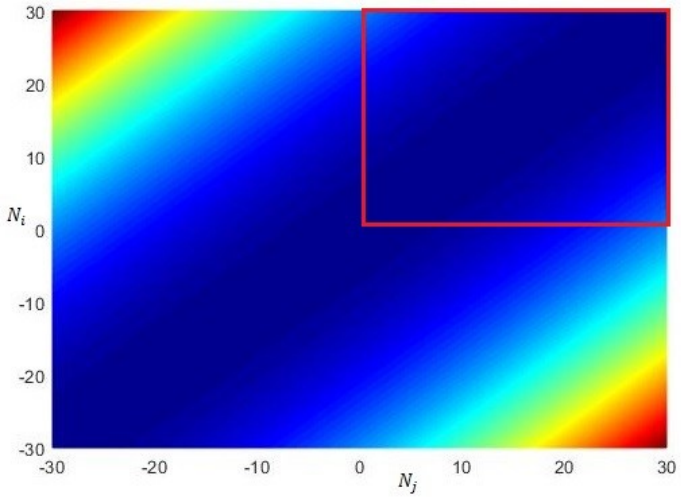


Figure 3.5: The contour of the shape of the function f with respect to N_i and N_j in case setting up the following parameters $m = 1, {s_i}^2 = 10, {s_j}^2 = 10$.

3.5 Computation complexity of the QEVSA

The QMSA is used as a computational infrastructure to reduce the computational complexity for selecting the optimum deployment scenario, the computational complexity of the quantum method of the entire system is $O\left(log_2(T){log_2}^3\left(\sqrt{d}\right)\right)$, it depends on the computational complexity of the QET function ${log_2}^3\left(\sqrt{d}\right)$ and the BSA embedded in quantum algorithm $log_2(T)$, where *d* refers to the total number of possible deployment scenarios such that the value of *d* according to the proposed RDM system equals *K*.

Setting up the maximum number of steps *T* is strongly influenced by the value of the minimum distance between variances of two scenarios α.

To set up the value of α, we need to compute all the distance values between variances of two different scenarios, then selecting the minimum distance between variances of two scenarios α as it is presented in Eq. (3.3). Here, the real problem lies in calculating properly the value α at every new incoming task (which means that the value of α changes for every new incoming task). Thus, the computational complexity of this operation is $O\left(\frac{K(K-1)}{2}\right)$ which is remarkably a challenging computational problem. So, to not confuse the reader, we will denote this repeatedly computed

value of α by α_{actual}. Finding the value of α_{actual} requires determining the minimum distance for any two different load distributions, mathematically expressed as $\alpha_{actual}(i,j) = \min_{\forall\, i,j} \left|(\sigma_i^2 - \sigma_j^2)\right|$.

Instead of calculating α_{actual} for each incoming task, the alternative solution is to compute only one time in advance the global non-zero minimum of α, denoted by α_{global}, i.e. computing the α_{global} before starting the operation distribution of the system. In compliance with Eq. (3.14), the expression of the value of α_{local} is influenced only by the load of changed computing units. Let's consider that the number of theoretical capacity resources type denoted by Z, as it is noticeable, in the almost of certain setup of computation of the different scenarios of σ^2, we obtain an identical value of $\sigma^2$for the computing units which have identical theoretical capacities, this means that computational complexity of α is closely related to the total number of theoretical capacity type Z. Therefore, taking into consideration the number of task type generators G, the computational complexity will be $O\left(\frac{ZG(ZG-1)}{2}\right)$, where $K \gg G$. Figure 3.6 summarizes the difference between the computational complexity for computing α using the classical method and the quantum one.

Unfortunately, there is a small drawback in the second method, if the value of α is fixed and very small, the value of the maximum number of steps during the BSA could be larger, but it will not increase significantly because we are using the logarithmic search.

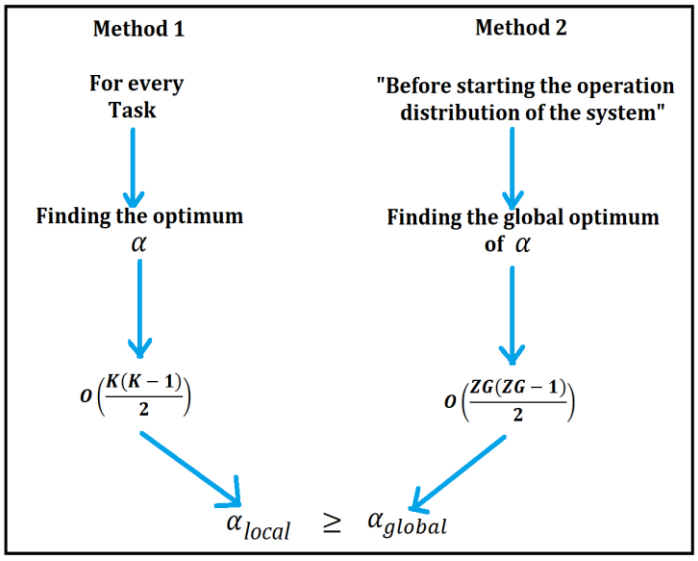


Figure 3.6: A straightforward explanation of the computational complexity in finding α for both methods, method 1 is the classical one and method 2 is the quantum one.

3.6 Simulation

To show that the QMSA outperforms other searching methods, we considered the randomized algorithm as a reference method. The aim is to compare the performance of the QEVSA with the randomized method. A simulation environment of Optimizer + Distributor was constructed. The model of a resource distribution system contains three processing units with different theoretical capacities, they have respectively the following capacities:

- The first computing unit has 20 as theoretical capacity.
- The second computing unit has 40 as theoretical capacity.
- The third computing unit has 60 as theoretical capacity.

Practical systems contain significantly more computing units, however, to observe trends and effects it is worthwhile investigating a small-scale model first. Furthermore, two task generator blocks are present in the system. The tasks are considered to have exponential arrival times. An Optimizer + Distributor block is considered to be a decision-maker. The length of task generation is fixed and it is equal to 20. In addition, all computing units have identical processing rate. The memory requirement and the service time of each task type are given by the following Table 3.1.

Table 3.1: Task type characteristics: service time and memory requirement.

	Task type 1	**Task type 2**
Memory requirement	3	2
Service time	1.2	0.9

The simulation is repeated for different task intensity distributions (In the proposed study, the exponential intensity distribution is considered). Table 3.2 presents the number of generated tasks for each intensity distribution.

Table 3.2: The number of generated tasks in each intensity distribution (length of time generation is 20).

	Mean = 0.5	**Mean = 0.4**	**Mean = 0.3**
Number of tasks	26	39	57

Moreover, the simulation environment does not take into account the case where the system is overload such that the load of the system can reach approximately 90%.

For each two different intensity distributions, the simulation is repeated 10 times, then the average relative load and the variance of the relative load for both algorithms are calculated i.e. the optimized and randomized methods.

As it is shown in Figure 3.7. (b) and Figure 3.8. (b), comparing the performance of the two distribution strategies, we sum up that although the changes in the average relative load, the variance of the proposed quantum solution remained approximately linear and tends to zero. This means that the quantum system conserves the distribution uniformity, while the variance of the reference (randomized) system follows the changes of the average relative load and diverges significantly from zero.

According to Figure 3.7. (a) and Figure 3.8. (a), the average relative load of the different methods remains approximately similar independently with the intensity of the exponential distribution of the arrival time of tasks. Furthermore, it is noticeable that when the intensity distribution of tasks increases, the randomized strategy diverges completely and fluctuates dramatically, while the optimized strategy keeps and conserves the load uniformity of the system. These results are illustrated in Figure 3.7. (b) and Figure 3.8. (b).

Last but not least, these simulation results assert that the quantum approach guarantees an efficient uniformity deployment of the system for any type of intensity distribution and low computational complexity.

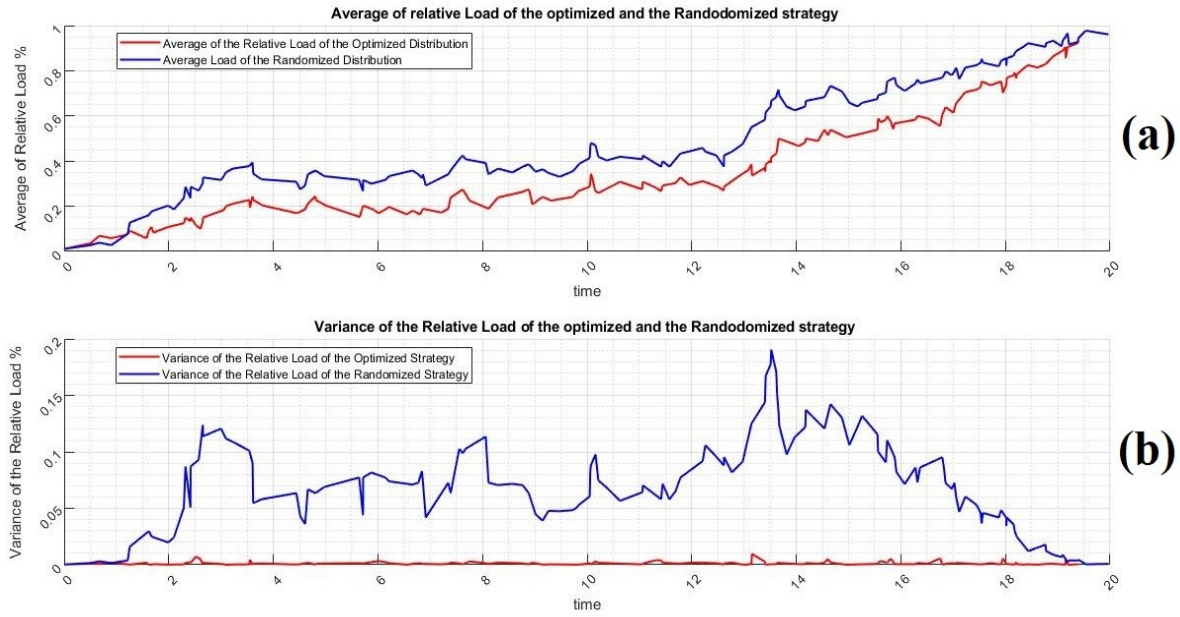


Figure 3.7: (a) The average relative load of the optimized strategy (red) and the randomized one (blue). (b) The variance of the relative load of the optimized strategy (red) and the randomized one (blue). In case the mean (intensity) of the exponential distribution of the arrival time of task type 1 is 0.3 and task type 2 is 0.4.

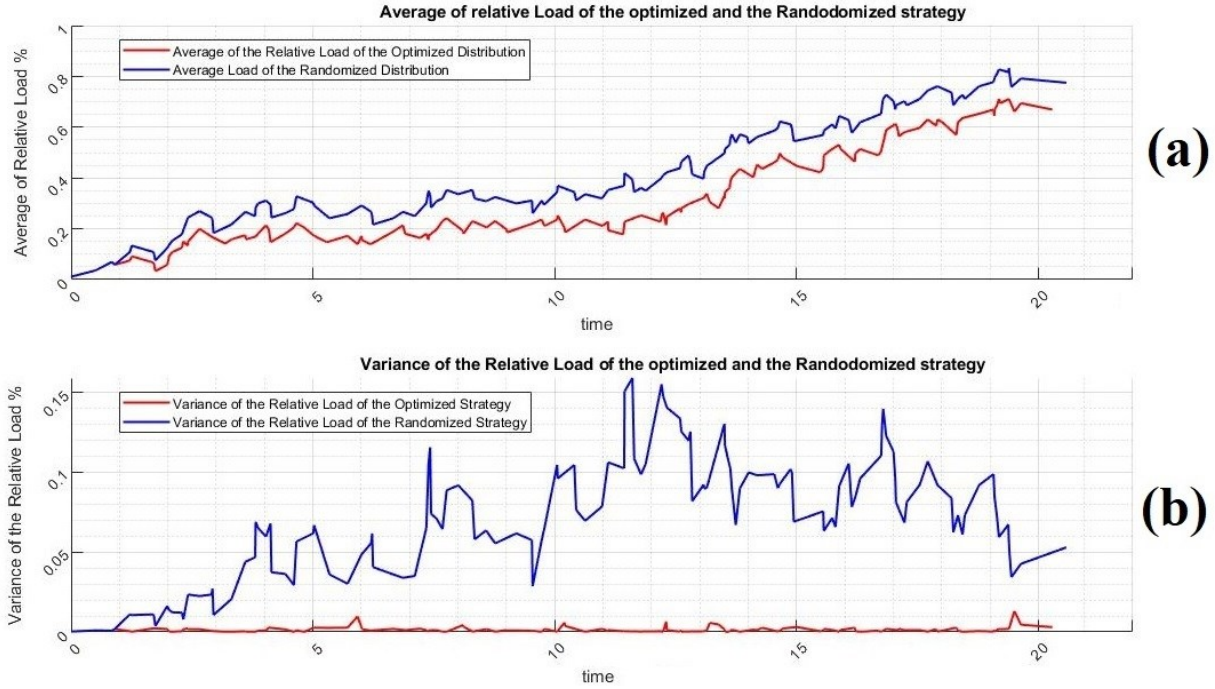


Figure 3. 8: (a) The average relative load of the optimized strategy (red) and the randomized one (blue). (b) The variance of the relative load of the optimized strategy (red) and the randomized one (blue). In case the mean (intensity) of the exponential distribution of the arrival time of task type 1 is 0.3 and task type 2 is 0.5.

3.7 Conclusion

A generalized model for the RDM system while considering multiple task types has been built. Following that, we showed how the QEVSA can be applied, as a computational infrastructure for RDM. Next, we configured the stochastic parameter T of the quantum strategy by taking into account the restrictions of the proposed RDM for multi-tasks.

Additionally, we showed that the computational complexity of the proposed quantum method can provide a significant reduction in the term of computational complexity and a high level of accuracy.

Finally, the gain of the quantum algorithm against the randomized reference has been demonstrated, this is done by extensive simulations which showed that whatever the intensity distribution, the quantum approach guarantees uniform distribution of tasks as well as a reduction in computational complexity.

4 Quantum optimization algorithms of resource distribution management for multi-tasks, multi-subtasks

With the significant growth and consumption of services provided. It is often difficult to predict the impact of the behaviour of a large amount of resources connected to a specific central of treatment. At the present, although the huge proposed combinatorial methods and evolutionary algorithms to overcome the planning task problem and choosing the best scheduling infrastructure of resource distribution management still present a bottleneck.

4.1 Introduction

For a fixed memory requirement, i.e. the contracted capacity amount used for the task, any available capacity left unreserved cannot be reused by other tasks. Maximizing and improving resource utilization requires using the multi-subtasks paradigm, i.e. any task consists of several subtasks. For this purpose, we extended our previous RDM model discussed in Chapter 3.

The rest of this chapter is organized as follows. Section 4.2 describes the proposed RDM model for multi-tasks, multi-subtasks. Section 4.3 studies the implementation and configuration of the QEVSA in the extended version of the RDM system. Section 4.4 investigates the computational complexity of the QEVSA in the case of handling multi-tasks and multi-subtasks case. Finally, Section 4.5 validates the efficiency of the QEVSA by using simulation results, and Section 4.6 concludes the chapter.

4.2 Description of RDM system for multi-tasks, multi-subtask

Let's consider multiple task generators, each of them releases a task type, each task type l is made of several subtasks selected from a set of subtask types, where the total number of different subtask types is W. The memory requirement for the subtask type v is Δ_v. Figure 4.1 represents the new RDM structure for multi-tasks, multi-subtasks.

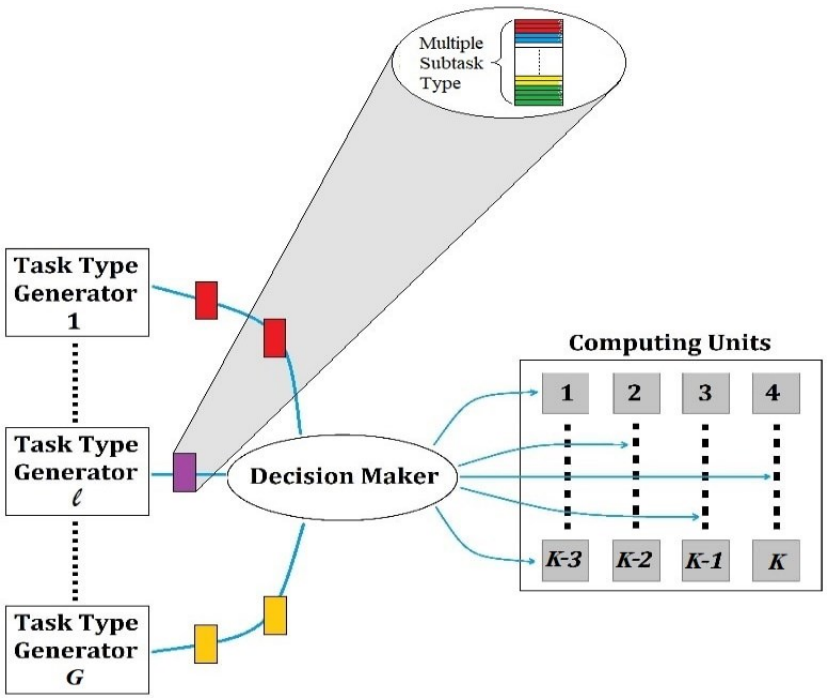


Figure 4.1: Resource distribution management model for multi-tasks, multi-subtasks case

4.3 Implementation and configuration of the QEVSA in RDM for multi-tasks, multi-subtasks

Similarly, to what has been discussed in Section 3.3, the QMSA algorithm will be adopted as a computational infrastructure for the new extended RDM model described above in Section 4.2.

As already mentioned, to fully exploit the potential of the QMSA, it is necessary to configure properly the parameters of the quantum method according to the characteristics of the RDM model.

As aforementioned, the parameter T depends on $\Delta\sigma^2$ as presented in Eq. (3.2). The possible region of the variance is bounded by 0 and 1. Also, the parameter T depends on α, one writes

$$\alpha = \min_{\forall V_i, V_j} \left| \sigma_{V_j}^2 - \sigma_{V_j}^2 \right|, \tag{4.1}$$

where V_i and V_j referred to two different assignment scenarios. For the sake of the minimum distance between variances of two scenarios α_{local}, so, it is necessary to find a suitable expression of $\left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)$.

Let x_k, b_k and $\bar{b}$ be respectively the total load, the relative load, and relative load average of the k^{th} computing unit before receiving the new subtask types, as it is shown respectively in Eq. (4.2), Eq. (4.3) and Eq. (4.4),

$$x_k = \sum_{v=1}^{V} N_{kv}^{b} \, \Delta_v \ , \tag{4.2}$$

where N_{kv}^{b} refers to the number of subtask type v in the k^{th} computing unit before deploying the new incoming task,

$$b_k = \frac{x_k}{s_k} \ , \tag{4.3}$$

$$\bar{b} = \frac{1}{K} \sum_{k=1}^{K} b_k \tag{4.4}$$

The problem resides in formulating a general expression of the minimum distance between variances of two scenarios ($\sigma_{V_i}^2 - \sigma_{V_j}^2$), i.e. for a given distribution scenario that fits to assigning only one subtask to a given computing unit. For this purpose, we considered that the load status of the computing units before and after the task deployment are given respectively by $\boldsymbol{V}^b$ (Eq. (4.5)) and $\boldsymbol{V}^a$ matrices, where the rows represent the computing units and the columns denote the subtasks type. To find the expression of the minimum variance between two different scenarios, the load of the new subtasks type should be taken into account, let's denote the set of computing units receiving the new subtasks by S_{ch} and the remaining set of computing units by S_{un}, such that $S_{ch} \cup S_{un} = S_T$. For the sake of a simple notation that describes all the possible deployment scenarios of the new incoming task, i.e. the new subtasks. Let $\boldsymbol{V}_i$ ($\boldsymbol{V}$ is bold because it is a matrix, i is not bold, because it is an index) be the matrix which denotes the possible distribution scenario i, its rows represent the computing units and the columns denote the subtask types as it is expressed in Eq. (4.6),

$$\boldsymbol{V}^b = \begin{bmatrix} N_{11}^b & \cdots & \cdots & \cdots & \cdots & N_{1W}^b \\ \vdots & \ddots & \ddots & \cdots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ N_{K1}^b & \cdots & \cdots & \cdots & \cdots & N_{KW}^b \end{bmatrix} \tag{4.5}$$

$$\boldsymbol{V}_i = \begin{bmatrix} N_{11}^{V^i} & \cdots & \cdots & \cdots & \cdots & N_{1W}^{V^i} \\ \vdots & \ddots & \ddots & \cdots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ N_{K1}^{V^i} & \cdots & \cdots & \cdots & \cdots & N_{KW}^{V^i} \end{bmatrix} \tag{4.6}$$

The relation between the load status of the computing units before and after assigning the new incoming task to computing units is $\boldsymbol{V}^b + \boldsymbol{V}_i = \boldsymbol{V}^a$.

The relative load of the set of computing units receiving the new subtasks S_{ch} is denoted by $b_{k\in S_{ch}}^{\boldsymbol{V}^i}$ and the relative load of the remaining computing units $b_{k\in S_{un}}$, the $b_{k\in S_{ch}}^{\boldsymbol{V}^i}$ and $b_{k\in S_{un}}$ are expressed as follows

$$\begin{cases} b_{k\in S_{ch}}^{V^i} = b_{k\in S_{ch}} + \boldsymbol{P}\boldsymbol{V}_i\Delta \\ b_{k\in S_{un}} = b_k \end{cases} \tag{4.7}$$

Where $b_{k\in S_{ch}}$ is the relative load of the set of computing units that are subject to receive the new subtasks (it refers to relative load without the new incoming task). Taking into consideration the new deployment subtasks, the average of the relative load of the RDM model is expressed by Eq. (4.8), note that $\boldsymbol{P} = \begin{bmatrix} \frac{1}{s_1} & \dots & \frac{1}{s_W} \end{bmatrix}$ refers to the theoretical capacity row vector and $\Delta = [\Delta_1 \quad \dots \quad \Delta_W]^t$ is the memory requirement of the subtask type (column vector), the symbol *t* refers to the transpose symbol.

$$\overline{b_{V_i}} = \frac{1}{K}\sum_{k=1}^{K} b_k + \frac{1}{K}\,\boldsymbol{P}\boldsymbol{V}_i\Delta. \tag{4.8}$$

The variance of the relative load of the new deployment scenario is given as follows,

$$\sigma_{V_i}^2 = \frac{1}{K}\sum_{k=1}^{K}\left(\overline{b_{V_i}} - b_k^{V_i}\right)^2. \tag{4.9}$$

In the end, we end up with the corresponding formula, which is expressed as follows,

$$\sigma_{V_i}^2 - \sigma_{V_j}^2 = \frac{1}{K}\left[\sum_{k\in S_{ch}^{V^i}}\left(\frac{\sum_v^W N_{kv}^{V^i}}{s_k}\right)^2 - \sum_{k\in S_{ch}^{Vj}}\left(\frac{\sum_v^W N_{kv}^{V^j}}{s_k}\right)^2 + 2\left(\sum_{k\in S_{ch}^{V^i}}\left(\sum_{v=1}^{W} N_{kv}^b\right)\left(\frac{\sum_v^W N_{kv}^{V^i}}{{s_k}^2}\right) - \sum_{k\in S_{ch}^{Vj}}\left(\sum_{v=1}^{W} N_{kv}^b\right)\left(\frac{\sum_v^W N_{kv}^{V^j}}{{s_k}^2}\right)\right)\right]. \tag{4.10}$$

- $N_{kv}^{V^i}$ and N_{kv}^{Vj} are respectively the number of subtask type v in the k^{th} computing unit after deploying the new incoming task according to scenarios V_i and V_j.
- $S_{ch}^{V^i}$ and S_{ch}^{Vj} are respectively the set of computing units receiving the new subtasks of scenario V_i and V_j.

The value of α denotes the smallest distance between two different scenarios among all the possible scenarios in a database. The α is illustrated in Figure 4.2.

It is important to mention that for any distribution, the difference between variances of two different scenarios depends only on the set of the computing units S_{ch} that have been assigned a workload, not on all computing units.

To find the minimum places of α, it is enough to investigate $\left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)^2 = 0$, this expression derives an important property of the minimum points (i.e., the variables N_{kv}^b, $v \in \{1, \dots, W\}$ are linearly dependent), in compliance with this result we conclude that the minimum places of the function $\left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)^2$ are situated in a hyperplane.

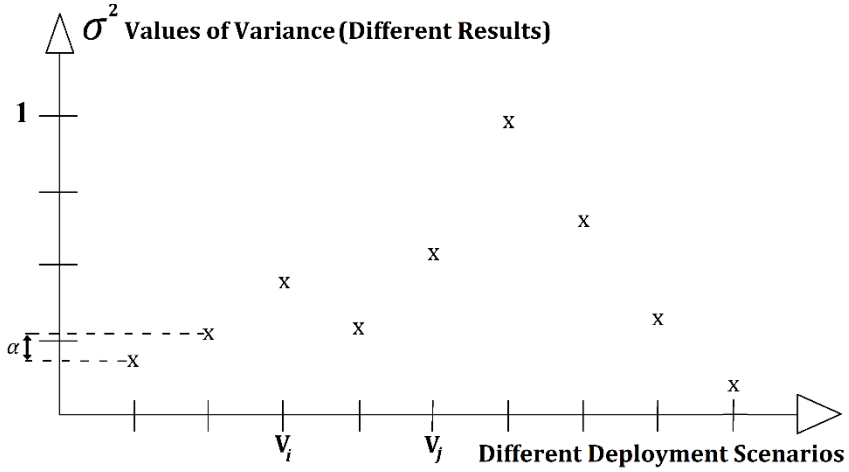


Figure 4.2: The horizontal axis presents all the possible deployment scenarios, while the vertical axis presents the borders of the variance function (different results), each possible scenario corresponds to a variance value. Computing the value of α requires selecting the minimum distance between variances of two deployment scenarios V_i and V_j.

In the general case, to determine the desired α, it is needed to fulfill a certain number of restrictions determined by choosing the suitable range of the variables N_{kv}^b, $v \in \{1, \dots, W\}$. The function $\left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)$ is continuous, however, from RDM point of view the variables N_{kv}^b, $v \in \{1, \dots, W\}$ must be integers. The first step is to assign an integer number different than zero to all the variables except one value, which corresponds to the subtask $v = r$, this value could be N_{kr}^b / $k \in S_{ch}^{V^i}$ (or $k \in S_{ch}^{V^j}$), then computing the value of N_{kr}^b / $k \in S_{ch}^{V^i}$ (or $k \in S_{ch}^{V^j}$), that corresponds to the remaining variables, here, at this stage, we have two options whether the value of N_{kr}^b / $k \in S_{ch}^{V^i}$ (or $k \in S_{ch}^{V^j}$) is an integer or not.

- 1st case: if N_{kr}^b / $R \in S_{ch}^{V^i}$ (or $R \in S_{ch}^{V^j}$) is an integer

The minimum distance between the variances of two scenarios α equals zero and the values of the variables N_{kr}^b, $v \in \{1, \dots, W\}$ are not appropriate solutions for α, so, in this case, we can modify N_{kr}^b / $k \in S_{ch}^{V^i}$ (or $R \in S_{ch}^{V^j}$), by increasing or decreasing N_{kr}^b, and assigning the value of $N_{kr}^b + 1$ or $N_{kr}^b - 1$ to N_{kr}^b / $k \in S_{ch}^{V^i}$, (or assigning the value of $N_{kr}^b + 1$ or $N_{kr}^b - 1$ to N_{kr}^b / $k \in S_{ch}^{V^j}$).

The most important thing is to choose only one assignment which results in the minimum value of α, let us investigate the value of $f = \left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)^2$ when we assign the value of $N_{kr}^b + 1$

or $N_{kr}^{b} - 1$ to N_{kr}^{b} , (or assigning the value of $N_{kr}^{b} + 1$ or $N_{kr}^{b} - 1$ to N_{kr}^{b} / $k \in S_{ch}^{V^j}$), following the same strategy adopted in Chapter 3 for proving Eq. 3.16. One obtains the following equality,

$$f_{N_{kr}^{b}+1} = f_{N_{kr}^{b}-1}. \tag{4.11}$$

The above expression states that the cross-section of the hyperplane is symmetric on the minimum places of $\left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)^2$ in its dimensional space.

- 2nd case: if N_{kr}^{b} / $k \in S_{ch}^{V^i}$ (or $R \in S_{ch}^{V^j}$) is not an integer

The function $\left(\sigma_{V_i}^2 - \sigma_{V_j}^2\right)^2$ has monotonous nature, consequently, the solution is to assign to N_{kr}^{b} / $k \in S_{ch}^{V^i}$ (or $k \in S_{ch}^{V^j}$) the nearest integer, in this case, α does not equal zero.

4.4 Computational complexity of the QEVSA

As previously presented, in order to reach a load balance, we have exploited the QMSA as a computational infrastructure for the RDM. Furthermore, we have proven that the computational complexity of the implemented QEVSA is $O\left(log_2(T) {log_2}^3\left(\sqrt{d}\right)\right)$, it depends on the computational complexity of the QET function ${log_2}^3\left(\sqrt{d}\right)$ and the BSA of the quantum algorithm $log_2(T)$, where *d* refers to the total number of possible deployment scenarios.

The computational complexity analysis is divided into two main parts,

It is interesting to note that Section 4.4.1. is devoted to estimating the search space *d*. While Section 4.4.2 investigates the computational complexity of determining the parameter *T*.

4.4.1 Estimation of the search space *d*

Let us assume that the new task under decision has arrived from generator g. This task contains M_{gv} identical subtasks from type *v* and we need to select M_{gv} pieces of computing units from the overall *K* where repetition is allowed, i.e. a certain computing unit can be chosen more than once. One can verify that the number of such possible different sets can be written as

$$\binom{K + M_{gv} - 1}{M_{gv}}. \tag{4.12}$$

Considering all the subtask types, one gets

$$d = \prod_{v=1}^{W} \binom{K + M_{gv} - 1}{M_{gv}}. \tag{4.13}$$

Now, we are ready to investigate the computational complexity of the lower and upper bounds of the size of search space *d*. It is easy to show that

$$\begin{aligned} \binom{K + M_{gv} - 1}{M_{gv}} &= \frac{(K + M_{gv} - 1)!}{(K-1)!\ M_{gv}!} \\ &= \frac{(K - 1 + M_{gv})!}{(K-1)!\ M_{gv}!} \end{aligned} \tag{4.14}$$

Assuming that $K \gg M_{gv}$, it is interesting to note that by using formula Eq. (4.14), one can verify that it can be expressed in the following manner

$$\begin{aligned} K\frac{(K+1)\cdot(K+2)\cdot \ldots \cdot (K-1+M_{gv})}{1\cdot 2\cdot \ldots \cdot M_{gv}} &= \frac{K}{M_{gv}} \prod_{i=1}^{M_{gv}-1} \frac{K+i}{i} \\ &= \frac{K}{M_{gv}} \prod_{i=1}^{M_{gv}-1} \left(\frac{K}{i} + 1\right) \end{aligned} \tag{4.15}$$

Now we are in a position to give the upper bound for Eq. (4.12). Using Eq. (4.15), one can confirm that

$$\frac{K}{M_{gv}} \prod_{i=1}^{M_{gv}-1} \left(\frac{K}{i} + 1\right) \leq \frac{K^{M_{gv}}}{\min\limits_{v} M_{gv}}. \tag{4.16}$$

Using Eq. (4.16) and Eq. (4.13) the upper bound of the size of the search space *d* can be expressed as

$$\prod_{v=1}^{W}\binom{K+M_{gv}-1}{M_{gv}} \leq \left(\frac{K^{M_{gv}}}{\min\limits_{v} M_{gv}}\right)^{W} \tag{4.17}$$

Next, we are interested in a close lower bound for Eq. (4.13). It is easy to verify that

$$\prod_{v=1}^{W}\binom{K+M_{gv}-1}{M_{gv}} \geq \frac{K^{M_g}}{\left(\max\limits_{v} M_{gv}\,!\right)^{W}} = \left(\frac{1}{\max\limits_{v} M_{gv}\,!}\right)^{W} K^{M_g} \tag{4.18}$$

where, $M_g = \sum_{v=1}^{W} M_{gv}$.

The lower bound expressed in Eq. (4.18) shows that the computational complexity of finding the optimal solution within the database is polynomial in terms of the numbers of computing units K but exponential in terms of M_{gv}, i.e. if the number of subtask type W becomes large, the computational complexity rises dramatically. For this sake, performing a classical optimization method to find the optimum result will be time-consuming and hard to solve. So, our proposed QEVSA is the best candidate to handle such a task assignment optimization problem.

4.4.2 Computational complexity of calculating α

The real problem in computing T lies in calculating properly the value of α at every incoming task (which means that the value of α changes for every new coming task), so, to not confuse the reader we will denote this repeatedly computed value of α by α_{actual}. Finding the value of α_{actual} requires determining the minimum distance for any two different load distributions, mathematically expressed as $\alpha_{actual}(V_i, V_j) = \min\limits_{\forall V_i, V_j} \left|\sigma_{V_i}^2 - \sigma_{V_j}^2\right|$. The computational complexity for finding α_{actual} is $O\left(\frac{d(d-1)}{2}\right)$, in the worst case, $d = c^{g_l}$, where g is the number of subtasks of the incoming task which is generated from type l, it is noticeable that computing the value of α_{actual} is computationally hard.

Instead of calculating α_{actual} for each incoming task, the alternative solution is to compute in advance the global non-zero minimum of α, denoted by α_{global}, before starting the operation of the system, such that $\alpha_{global} = \min\limits_{\forall S_{ch}^i, S_{ch}^j} \alpha_{local}(S_{ch}^i, S_{ch}^j)$, where α_{local} means minimization over

all possible load configurations of changed unit sets S_{ch}^{i} and S_{ch}^{j} belonging to the distributions V_i and V_j. Because a certainly changed unit set fits many distributions, it is enough to define the previous formula with i and j. The expression of the value of α_{local} is influenced only by a load of changed computing units, as expressed in Eq. (4.10), while the unchanged computing units have no effects on α_{local}.

Investigating all unit set pairs to calculate α_{global}, we need to compute all α_{local} which are related to unit sets, assuming that the number of computing unit types is θ, taking advantage of the previous statement of α_{global}, we conclude that the number of possible distributions d' is less or equal than $\theta^{g_{max}}$ where g_{max} is the number of arrival subtasks, which means that the computational complexity at this stage is $O\left(\frac{d'(d'-1)}{2}\right)$, this computation complexity is significantly less than the computation complexity of α_{local}, the disadvantage of this minimization will create an increase in the maximum number of steps, but the quantum approach can handle the logarithm complexity of a large number of scale values because it will not increase significantly the complexity of the system.

4.5 Simulation results

The simulation environment hosts two task-type generators. We consider three subtask types and two decision-makers, the first one is dedicated to the optimized strategy and the second one is devoted to the randomized method. The description of the simulation environment is presented in Figure 4.3. The time slot for the generation of tasks is 10. The memory size of the subtask types is presented in Table 4.1. The number of subtask types in each task type is described in Table 4.2, while the characteristics of the computing units are presented in Table 4.3.

Table 4.1: The memory size of subtask types.

	Number of subtask type 1	Number of subtask type 2	Number of subtask type 3
Memory	3	2	4

Table 4 2: The number of subtask types in each task type.

	Number of subtask type 1	Number of subtask type 2	Number of subtask type 3
Task type 1	1	1	0
Task type 2	0	1	1

Table 4.3: Characteristics of computing units.

	Computing unit 1	Computing unit 2	Computing unit 3
s_k	20	40	60

We have constructed an experiment where the mean values of the exponential intensity distribution of each task generator are respectively 0.09 and 0.1, and the total number of generated tasks in each exponential intensity distribution are respectively 92 and 107.

According to Figures 4.4 and 4.5, It is noticeable that although the changes of the average relative load, the variance of the relative load of the optimized strategy tends to zero while the variance of the relative load of the randomized method fluctuates and diverges from zero. To sum up, the optimized strategy provides better results in terms of reaching the best uniformity distribution while the randomized one does not.

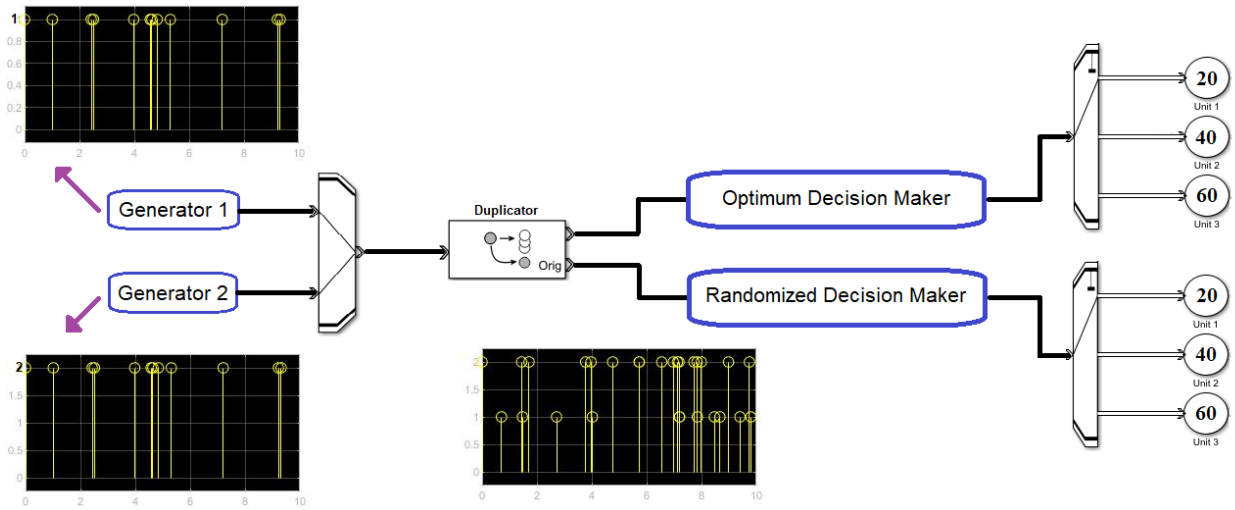


Figure 4.3: Simulation environment of multi-tasks, multi-subtasks

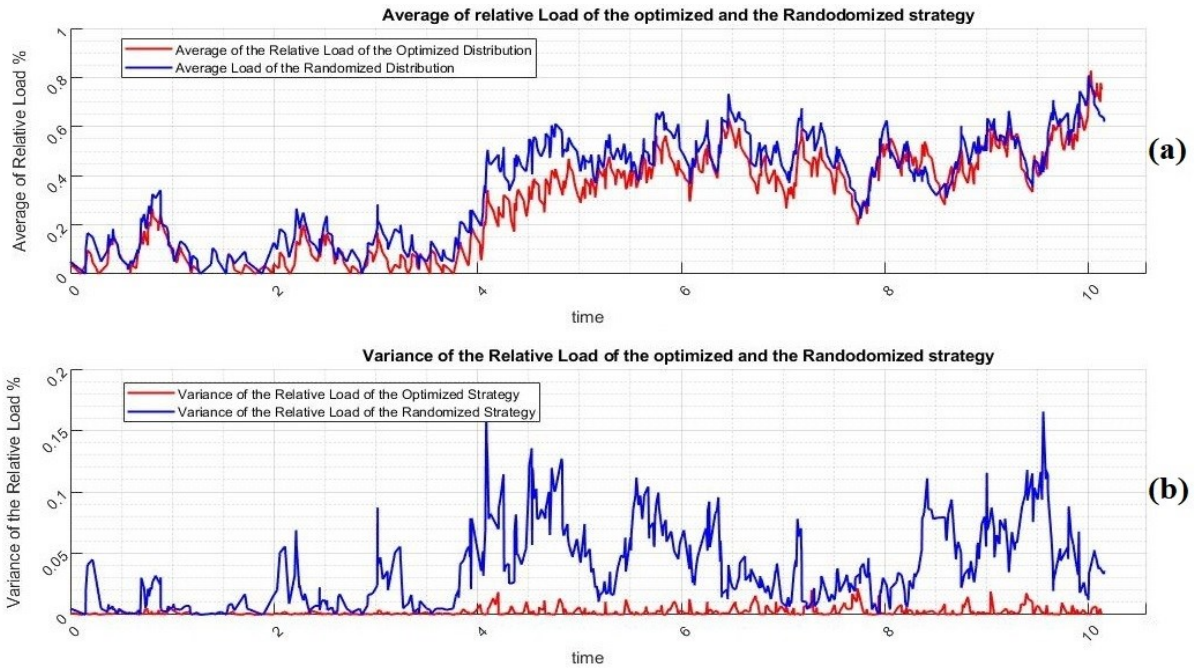


Figure 4.4: (a) The average relative load of the optimized strategy (red) and the randomized one (blue). (b) The variance of the relative load of the optimized strategy (red) and the randomized one (blue). In case the mean (intensity) of the exponential distribution of the arrival time of task type 1 is 0.1 and task type 2 is 0.09.

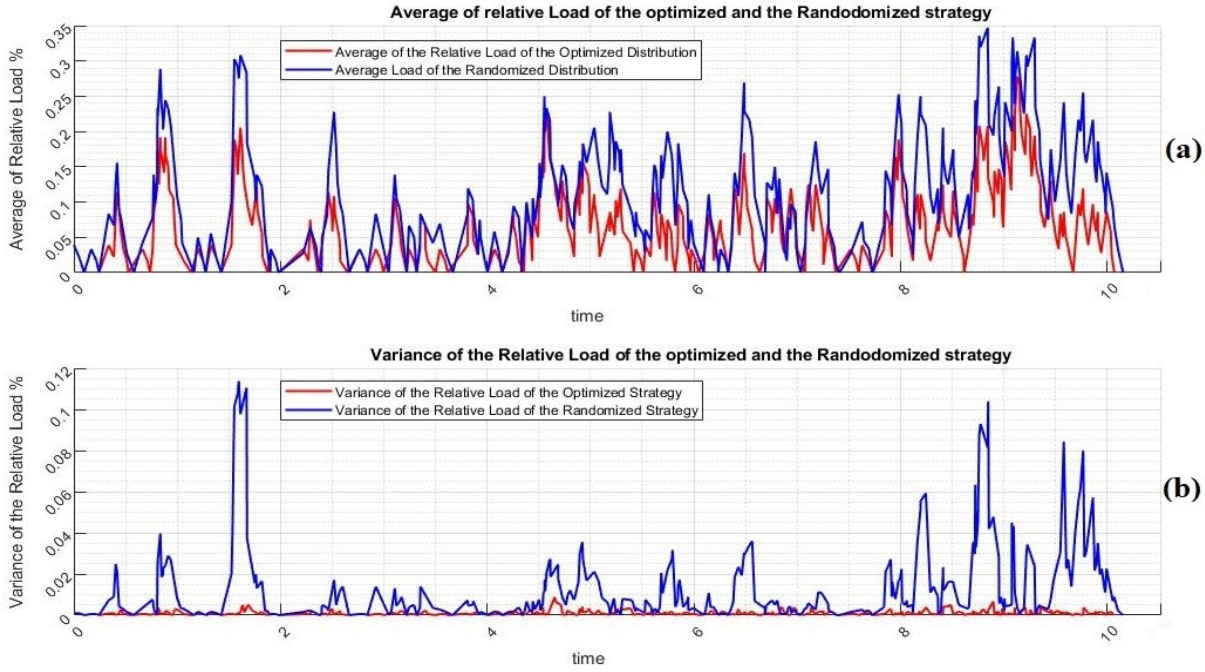


Figure 4.5: (a) The average relative load of the optimized strategy (red) and the randomized one (blue). (b) The variance of the relative load of the optimized strategy (red) and the randomized one (blue). In case the mean (intensity) of the exponential distribution of the arrival time of task type 1 is 0.2 and task type 2 is 0.1.

4.6 Conclusion

In this Chapter, we extended the RDM model described in Chapter 3, to multi-tasks, multi-subtasks case. Next, a measurement metric (the variance of the relative load) has been chosen to test the efficiency of the QEVSA. The set up of the stochastic parameter T of the QEVSA has been investigated. We showed that the value of T depends only on the set of computing units S_{ch} that have been assigned a workload, not on all computing units.

The total number of possible deployment scenarios for handling the multi-tasks, multi-subtasks case has been estimated and we proved that the proposed assignment problem cannot be solved through classical optimization methods, because the computational complexity of finding the optimal solution within the database is polynomial in terms of the numbers of the computing units but exponential in terms of the numbers of subtasks.

Simulation results showed that the optimized strategy provides better results in terms of reaching the best uniformity distribution while the randomized one does not.

5 Constrained Quantum Optimization Based Resource Distribution Management for Multi-Tasks, Multi-Subtasks

As it is known, reducing energy consumption and improving resource utilization by taking into account the constraints arising from the resource distribution management system is regarded as one of the central interests of the information technology sector. In order to support the goals of the quality of service and decision-making process, new computing methods must be implemented.

5.1 Introduction

In this Chapter, the problem of RDM is formulated as a constrained optimization problem. The energy consumption with respect to delay constraint of the RDM system, where the multi-tasks, multi-subtasks case is considered as a toy example for pointing out the efficiency and the strength of the proposed CQOA.

This chapter is organized as follows. Section 5.2 introduces an extension of the RDM system, as well as the problem formulation. Section 5.3 describes the implementation and configuration of the CQOA according to the new extended version of RDM. Section 5.4 discusses the computational complexity of the applied constrained quantum strategy. Section 5.5 concludes the paper by simulation results, where a comparison between the proposed constrained optimized method and the constrained randomized one is investigated, and Section 5.6 is a conclusion.

5.2 Description of the RDM system for multi-tasks, multi-subtasks in case of optimizing a constrained goal function

The goal is to reduce the energy consumption of computing units by taking into account the delay constraint of the tasks. We considered the RDM for multi-tasks, multi-subtasks presented in Figure 4.1.

Let's define the extended version of the RDM model for multi-tasks, multi-subtasks. Each generated task type g has to obey a specific delay constraint denoted by τ_g, i.e. all the subtasks of the task have to be served within this delay constraint.

The decision-maker deploys the subtasks of the incoming task to different computing units. Assuming that the subtasks of the incoming tasks are processed sequentially. Figure 5.1 illustrates the processing mechanism of subtasks in the k^{th} computing unit, where subtasks arrive in a FIFO manner, i.e. when a task arrives in the decision-maker, it decides instantly whether the task (consisting of subtasks) can be accepted and deployed to any of the computing units or not (rejected). The delay needed to process the actual load in the k^{th} computing unit (considering the task under the decision, too) denoted by τ_k^{act}, it has to be always less or equal than the delay constraint of the incoming task type, i.e. $\tau_k^{act} \leq \tau_g$. The green hatched subtask is the subtask belonging to the new incoming task deployed into the k^{th} computing unit (see Figure 5.1). This decision method guarantees that the delay constraint of tasks that are running on the k^{th} computing unit does not influence the fulfillment of delay constraint of the new incoming task and vice versa.

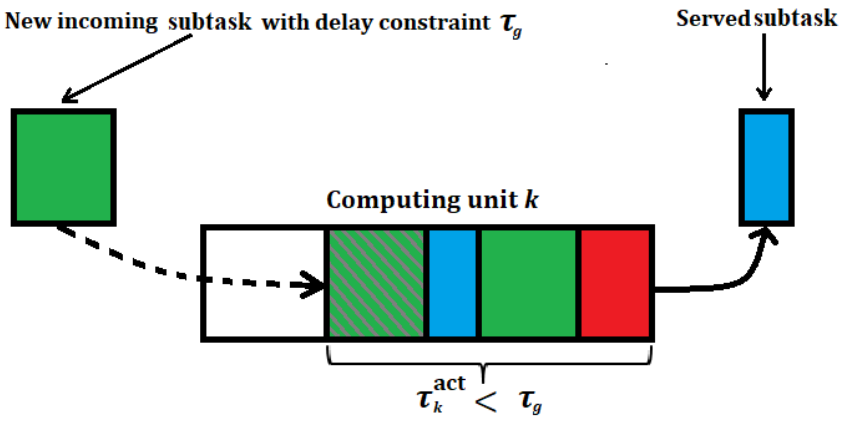


Figure 5.1: Scheme illustrating the sequential processing operation of different subtasks in the k^{th} computing unit.

Now, let's investigate the calculation of the total energy consumed by the RDM. We consider that the initial power needed to turn on the computing unit k is denoted by P_k^{init}.

As we already mentioned, a subtask is composed of a specified number of identical memory pieces called memory units. The time needed to process a subtask type v on the computing unit k is

Δ_v/β_k. It is interesting to note that the processing delay of the actual load on the k^{th} computing unit can be calculated as,

$$\tau_k^{act} = \frac{\sum_{v=1}^{W} N_{kv}\Delta_v}{\beta_k} \tag{5.1}$$

Assuming that $\tau_k^{act} \leq \tau_g$ and the processing of the subtasks is performed sequentially as illustrated in Figure 5.1, the energy required to process the subtasks on computing unit *k* is given by Eq. (5.2), where ε_k is the energy consumption of one memory unit on computing unit *k*.

$$E_k^{act} = \varepsilon_k \sum_{v=1}^{W} N_{kv}\Delta_v + P_k^{init}\tau_k^{act} = \left(\varepsilon_k + \frac{P_k^{init}}{\beta_k}\right)\sum_{v=1}^{W} N_{kv}\Delta_v, \tag{5.2}$$

- $\varepsilon_k \sum_{v=1}^{W} N_{kv}\Delta_v$: The energy needed to serve the subtasks under process on the computing unit *k* without considering the energy required to turn on the computing unit *k*.

- $P_k^{init}\tau_k^{act}$: The energy needed to turn on the computing unit *k* in a period that is equal to τ_k^{act} (the energy of tasks is not considered).

It is straightforward to verify that the overall energy consumption of the system can be written as,

$$\begin{aligned} E^{act} &= \sum_{k=1}^{K} E_k^{act} \\ &= \sum_{k=1}^{K}\left(\varepsilon_k + \frac{P_k^{init}}{\beta_k}\bigg|_{k\in H^{on}}\right)\sum_{v=1}^{W} N_{kv}\Delta_v. \end{aligned} \tag{5.3}$$

Where the term $\frac{P_k^{init}}{\beta_k}$ is considered if and only if the k^{th} computing unit is switched on (The set H^{on} of the switched-on computing units). Our goal is to minimize the overall energy consumption E^{act}, one obtains

$$E_{min}^{act} = \min_{\forall S_i \in S} E^{act}(S_i) \; and \; \forall k \in S_i : \tau_k^{act} \leq \tau_g \tag{5.4}$$

with S denotes the sets of different possible distributions of the subtasks of the incoming task among all computing units and S_i refers to the i^{th} distribution/deployment scenario of S, i.e. the i^{th} specific set of computing units $S_i = \{k\}$.

The next section will be devoted to explaining the implementation and configuration of the CQOA in the proposed RDM model.

5.3 Implementation and configuration of the CQOA

To select the best and the optimum deployment scenario, we apply the CQOA as a minimum constrained searching algorithm (MCSA). To this end, the function F is substituted by E^{act}, while the constraint C corresponds to the delay constraint of the incoming task type i.e. τ_g, and the implemented relation R is a "minimization".

It is worth mentioning that it is enough to investigate the delay constraint of the new task, because all former delay constraints will be met (Tasks are processed sequentially inside the computing units.

The maximum number of steps needed to run the logarithm search T expressed Eq. (5.5) depends on two parameters: the variation of the energy consumption of the system denoted by $\Delta E = E_{max}^{act} - E_{min}^{act}$ and the step size α which is the smallest distance between the energies of two different scenarios among all the possible scenarios in the database. Figure 5.2 shows these parameters.

Since the searching region is obviously $\Delta E = E_{max}^{act} - E_{min}^{act}$, the stochastic parameter T can be expressed as follows

$$T = \frac{E_{max}^{act} - E_{min}^{act}}{\alpha}, \tag{5.5}$$

where E_{min}^{act} can be replaced by the energy consumption of the system without the new incoming task. We denote this energy value by $\breve{E}^{act}$ and one can observe that it does not depend on the deployment of the incoming task, this implies that

$$T = \frac{E_{max}^{act} - \check{E}^{act}}{\alpha} \ . \tag{5.6}$$

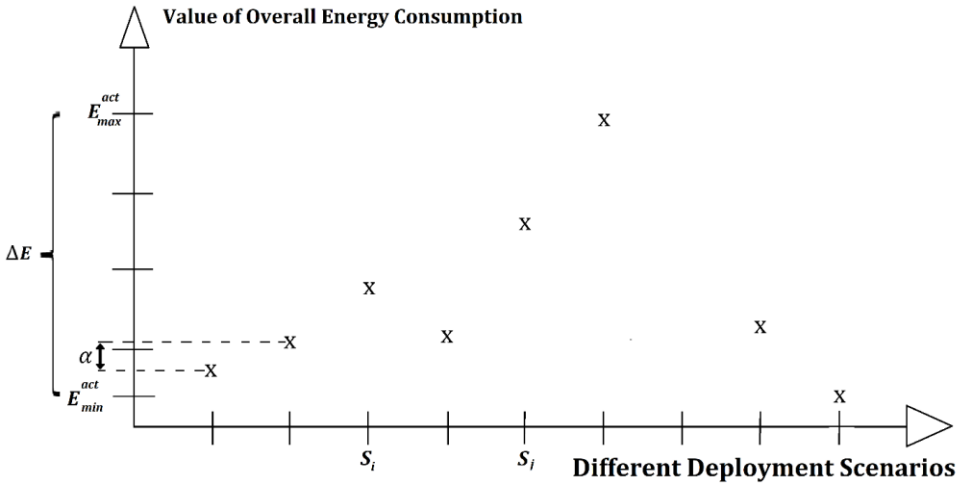


Figure 5. 2: The horizontal axis presents all the possible deployment scenarios, while the vertical axis presents the borders of the total energy consumption function (different results), each possible scenario corresponds to total energy consumption. Computing the value of α requires selecting the minimum distance between total energy consumption functions of two deployment scenarios S_i and S_j.

An appropriate worst-case estimation for E_{max}^{act} can be considered as

$$E_{max}^{act} = \check{E}^{act} + \tau_g \sum_{k \in H^{OFF}} P_k^{init} + E_{max}^{inc}, \tag{5.7}$$

Where,

- E_{max}^{inc} denotes the energy consumption if the subtasks of the incoming task are deployed onto that computing units having the largest ε_k, i.e. the least energy-efficient unit.
- $\tau_g \sum_{k \in H^{off}} P_k^{init}$ is the total energy consumed by the set H^{off} of the switched-off computing units because the switched-om computing units are already considered in $\check{E}^{act}$.

On the other hand, the value of α can be written as,

$$\alpha = \min_{\forall\, i \neq j} \left| E^{act}(S_i) - E^{act}(S_j) \right|, \tag{5.8}$$

where $E^{act}(S_i)$ is the sum of the energy consumption of the system without the new task $\breve{E}^{act}$ and the energy consumption of the incoming task with assuming that it was distributed according to Z_i,

$$E^{act}(S_i) = \breve{E}^{act} + E^{inc}(Z_i), \tag{5.9}$$

where Z_i refers to the i^{th} set of those computing units which receive one or more subtasks of the new incoming task. It is straightforward to verify if one substitutes Eq. (5.9) into Eq. (5.8), one obtains

$$\alpha = \min_{\forall\, i \neq j} \left| E^{inc}(Z_i) - E^{inc}(Z_j) \right|, \tag{5.10}$$

Let M_{kvi} be the number of subtasks of the incoming task from subtask type *v* deployed onto computing unit *k* in case of the i^{th} deployment scenario. The formula of $E^{inc}(Z_i)$ can be expressed now by means of Eq. (5.3) as

$$E^{inc}(Z_i) = \sum_{k \in Z_i} \left(\varepsilon_k + \frac{P_k^{init}}{\beta_k}\bigg|_{k \in H^{on}} \right) \sum_{v=1}^{W} M_{kvi} \Delta_v, \tag{5.11}$$

Where this term $\frac{P_k^{init}}{\beta_k}$ is considered if and only if the k^{th} computing unit is switched on. To set up properly the stochastic parameter α, it is enough to investigate the non-zero solutions of $\left(E^{inc}(Z_i) - E^{inc}(Z_j)\right)^2 = 0$, the solutions are located on a hyper-plane, this result is already discussed in Chapters 3 and 4.

5.4 The computational complexity of the CQOA

As previously presented, in order to minimize the constrained overall energy consumption, we have exploited the CMSA as a computational infrastructure for the RDMS. Furthermore, we have proven that the computational complexity of the implemented CQOA is $O\left(log_2(T) {log_2}^3\left(\sqrt{d}\right)\right)$, it depends on the computational complexity of the CQRT function ${log_2}^3\left(\sqrt{d}\right)$ and the BSA $log_2(T)$, where *d* refers to the total number of possible deployment scenarios.

The computational complexity analysis is divided into two main parts,

- The size of the search space d which refers to the total number of possible deployment scenarios, the procedure of calculating the value of d is similar to what has been shown in Section 4.4.1. To that end, the value of d can be calculated as expressed in Eq. (4.13), and the lower bound of d is presented in Eq. (4.18). Therefore, the computational complexity of finding the optimal solution within the database is polynomial in terms of the numbers of computing unit K but exponential in terms of M_{gv}. We confirm that the size of the search space d for the proposed constrained optimization assignment problem is similar to the previous unconstrained optimization case. Thus the constrained classical optimization algorithms cannot handle this kind of constrained optimization problem.

- The value of T roughly depends on the value of the stochastic parameter α. As already discussed in Section 4.4.2, computing repeatedly the value of α poses a real challenge. For this sake, we proposed an alternative solution that consists of setting up the T value in advance before starting the assignment operation, in this case, the computational complexity of determining parameter α of the BSA will be $O(1)$. More details on how to configure the stochastic parameter T are presented in Section 4.4.2.

5.5 Simulation results

To demonstrate the efficiency of the proposed CQOA, a simulation environment has been constructed, in which a constrained randomized method was considered as a reference for comparison with the proposed constrained optimization method. In the best case, the computational complexity of the constrained randomized method is $O(const)$, and in the worst case, it is $O(d)$. It is interesting to note that this simulation aims to compare the performance of both methods in terms of computational complexity and the overall energy consumption with respect to the delay constraint.

The simulation of the RDMS hosts three computing units, the characteristics of computing units are presented in Table 5.1, we considered three computing units where they have an identical processing rate of 40. From a practical point of view, the RDMS contains significantly more

computing units, however, to observe the trends and effects it is worthwhile investigating a small-scale model.

Also, we considered two task type generators such that tasks are generated exponentially, where one of the task types has a high-intensity distribution (the mean value of intensity distribution is smaller) compared to the other one. Furthermore, we considered two subtask type, their memory requirements are respectively $\Delta_1 = 2$ and $\Delta_2 = 4$. Table 5.2 presents the number of subtask types, total memory required, and the delay constraint of each task type. While table 5.3 shows the total number of tasks released for each intensity distribution. It is important to mention that the timeslot of task generation is 20 seconds.

Table 5.1: Characteristics of computing units.

	Computing Unit 1	**Computing Unit 2**	**Computing Unit 3**
c_k	40	80	120
P_k^{init}	20	25	30
ε_k	1	2	3

Table 5.2: The characteristics of the generated task types.

	Number of Subtask type 1	**Number of Subtask type 2**	**Memory**	**Delay constraint**
Task type 1	40	80	120	1.4
Task type 2	20	25	30	1.2

Table 5.3: The number of tasks released for each intensity distribution

	Mean = 0.4	**Mean = 0.3**	**Mean = 0.2**	**Mean = 0.1**
Number of tasks	39	57	91	203

It is interesting to note that the decision-maker checks first the capacity constraint, i.e. if there is free space in the system for allocating the new task. Then, it checks the delay constraint. Also,

it is very important to mention that we did not deal with the case where the system is overloaded, i.e. we did not investigate the queueing behavior of the system.

For each two task types that have different intensity distributions, we repeat the simulation 10 times, then we calculate the average overall energy consumed for each algorithm.

Figure 5.3 compares the total energy consumed by the randomized and optimized methods for different exponential intensity distributions. It can be seen that for every experiment, the constrained optimized strategy consumes less energy than the constrained randomized one. Additionally, one can notice that when the intensity distribution of arrival tasks increases, i.e. when the mean value of exponential distribution becomes smaller, the optimized strategy keeps consuming lower than the randomized one.

Figure 5.4 shows the percentages of the overall energy consumption reduction of the three aforementioned experiments. For example, in the first experiment where the two task type generators have respectively 0.4 and 0.3 as means (the mean value of exponential distribution), the energy consumption of the optimized method is less than the randomized one by approximately 43.85 %.

As it is shown, in the worst case the constrained randomized algorithm uses d steps, but it cannot find the optimum deployment scenarios which correspond to the minimum total energy consumption of the RDMS. While the computational complexity of the constrained optimized strategy is $O\left(log_2(T){log_2}^3(\sqrt{d})\right)$.

To sum up, we see that whatever the distribution intensity is, the constrained optimized approach offers a significant reduction in terms of energy consumption and computational complexity.

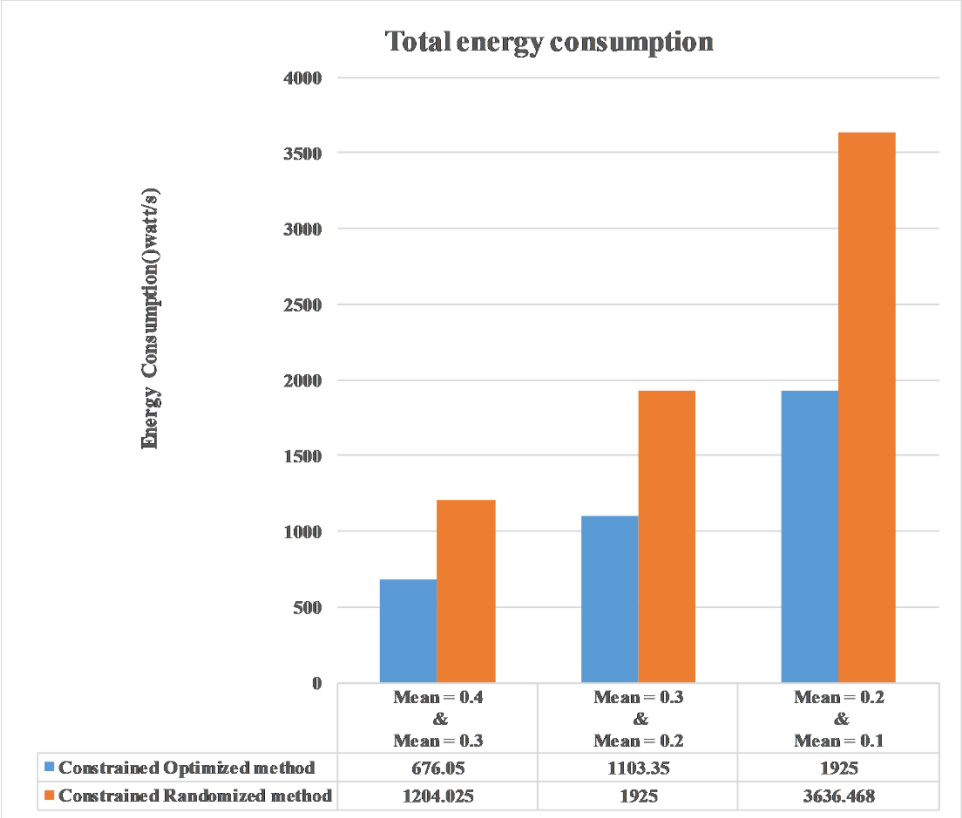


Figure 5.3: Energy consumption of the Optimized (Blue bars) and the randomized (Red bars) strategies according to different intensity distributions.

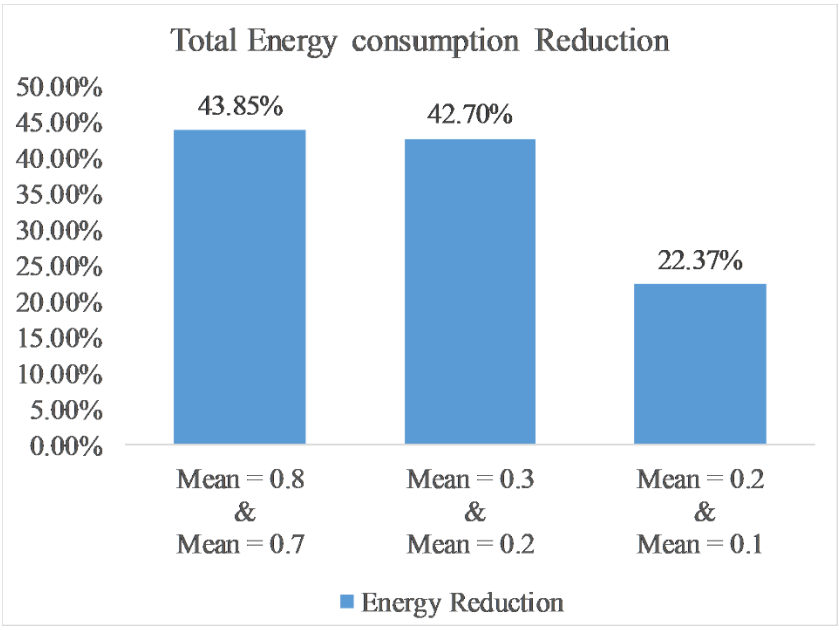


Figure 5.4: Energy consumption reduction.

5.6 Conclusion

In this Chapter, I exploited the CQOA to minimize the constraint goal function (The total energy consumption) of the RDMS. We derived a simplified form of the constraint goal function, and we investigated the implementation and the configuration of the proposed constrained quantum strategy. Next, we proved that the computational complexity of finding the optimal solution within the database is polynomial in terms of the numbers of the computing units but exponential in terms of the number of subtasks. Fortunately, the proposed CQOA can handle such kind of optimization problem exponentially faster. In the end, we demonstrated by a simulation environment the effectiveness of the CQOA in terms of energy consumption and computational complexity by making a comparison between the constrained randomized strategy and the constrained quantum one, where I proved that whatever the distribution intensity is, the optimized approach offers the best performance and can significantly reduce the energy consumption of computing units.

6 Constrained Quantum Optimization Based Resource Distribution Management with Considering Queueing Scenarios.

As it is known, the length of the waiting queue and quality of service are two aspects that are linked, this is the reason why optimizing and improving the queueing system process still a key focus in several application domains. For this sake, it turns out that the queueing behavior demands more computational power than the normal deployment system.

6.1 Introduction

In the present Chapter, we extend the result presented in Chapter 5, we implement the CQOA in the RDM model where the queueing scenarios of RDM of multi-task, multi-subtask are considered.

The rest of this chapter is organized as follows. Section 6.2 introduces the RDMS with considering queuing scenarios and the problem statement. Section 6.3 investigates the implementation and configuration of the CQOA in the new RDM system. Section 6.4 studies the computational complexity of selecting the optimal distribution scenario that corresponds to the minimum energy consumption. Section 6.5 shows the efficiency of the CQOA by constructing a simulation environment. Section 6.6 concludes the Chapter.

6.2 Description of RDM system with considering queueing scenarios

This subsection presents the RDM by considering one shared queue among computing units (see Figure 6.1), as well as, introduces the problem of minimizing the overall energy consumption with respect to the delay constraint of the incoming tasks.

We consider the previous RDM model described in Chapter 5.2.

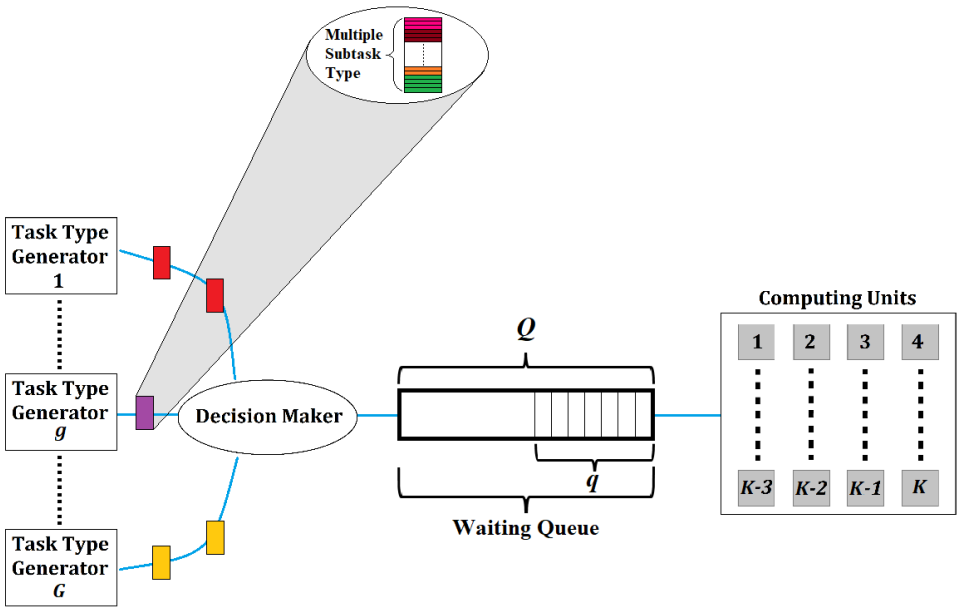


Figure 6.1: Resource distribution management system with a shared waiting queue

Let us assume that we have one free access waiting queue shared among computing units, i.e. the tasks in the waiting queue can be deployed in different manners without respecting the order of their arrival, where the queue has no energy consumption. Let Q denote the length of the queue in the memory unit and q the actual number of tasks in the waiting queue including the new task.

Also, assuming that the total generated tasks from different generators is D, they are organized according to their arrival time from 1 to D such that $D \in \mathbb{N}$.

It is important to note that the i^{th} incoming task released from generator g requires to process its subtasks within a delay constraint τ_g. To this end, Eq. (5.2) is transformed to

$$\tau_{i,k}^{act} = \frac{\sum_{v=1}^{W} N_{kv}\Delta_v}{\beta_k}. \tag{6.1}$$

It is interesting to emphasize that from queueing delay constraint point of view. There are two main optimization delay approaches, either we consider the actual processing delay plus the waiting delay as one global delay constraint such that

$$\forall\, k \quad \theta_i + \tau_{i,k}^{act} \leq \tau_g, \tag{6.2}$$

where θ_i is the waiting delay of the i^{th} task in the free access waiting queue (coming from the waiting queue). Or, we assume that there are two different delay constraints, the first one belongs to the processing delay constraint of the incoming task type while the second one belongs to the queueing delay constraint of the same incoming task type, such that

$$\forall\, k \quad \begin{cases} \tau_{i,k}^{act} \leq \tau_g \\ \theta_i \leq \theta_q \end{cases}, \tag{6.3}$$

where θ_q is the maximum time permitted for a given task type to stay in the waiting queue. In the sequel, we assume that the proposed RDM follows the first constraint optimization delay approach Eq. (6.2).

On the other hand, we always need to check the capacity constraint of the system first, then we verify the delay constraint. The working mechanism of task deployment is defined as follows,

- If there is free space in the computing units and the delay constraint of the incoming task is met, the subtasks derived from the incoming task are distributed to the optimum places.
- If there is free space in computing units and the delay constraint of the incoming task is not satisfied, the arrived task is rejected.
- If there is no free space in the computing unit, the incoming job is sent back to the free access waiting queue.

It is interesting to emphasize that when there is a free capacity space in the system, and there are tasks in the waiting queue, the selection of the next task is performed according to the task that has the most priority to be deployed. Here, we consider the situation where the deployment priority is given to the tasks which have the largest global delay, this delay is the sum of the waiting time in the queue and the time required to serve the task.

If there is free capacity space in the computing units for the new i^{th} incoming task released from generator g, then, it is enough to $\theta_i = 0$, while if the i^{th} task of generator g is coming from the waiting queue and there is free capacity space in the system, then $\theta_i \neq 0$.

The goal is to minimize the total energy consumption of the RDMS with respect to the delay constraint of the incoming tasks. For this reason, we apply the CQOA as a computational method for the decision-maker.

The optimized decision-maker chooses the optimum possible computing unit set which satisfies the delay constraint and which corresponds to the minimum total energy consumption. Assuming that there is free capacity space in the k^{th} computing unit and the delay constraint holds. The energy needed to serve the subtasks on computing unit *k* is expressed as follows

$$E_{min}^{act} = \min_{\forall S_i \in S} E^{act}(S_i) \; and \; \forall k \in S_i \qquad \theta_i + \tau_{i,k}^{act} \leq \tau_g. \tag{6.4}$$

6.3 Implementation, configuration, and computational complexity of the CQOA with considering queueing scenarios

The CQOA is applied as an embedded computational tool in the decision-maker, where the decision-maker selects the optimum distribution scenario that fits the minimum constrained overall energy consumption of RDMS. Therefore, the CQOA is implemented as a constrained minimum searching algorithm. The implemented relation R is less or equal ($\leq$) (see Table 2.1), while the function F will be substituted by the overall energy consumption function E^{act}, i.e. $F = E^{act}$. The constraint *C* corresponds to the delay constraint of task type τ_g.

The computational complexity of the CQOA is $O\left(log_2(T) {log_2}^3\left(\sqrt{d}\right)\right)$. It depends on the computational complexity of,

- The BSA embedded in the CQOA $log_2(T)$, where the configuration methodology of the stochastic parameter *T* of the CQOA is similar to the one belonging to the QEVSA.
- The CQRT function ${log_2}^3\left(\sqrt{d}\right)$, where *d* refers to the entry size of the database (search space). Here, in the proposed studied case, the variable d refers to the overall possible deployment scenarios for each new incoming task.

Estimating the search space *d* in the non-waiting queue case differs from the case where we have tasks sent to the waiting queue. We already investigated the search space *d* of the non-waiting

queue case in Chapter 4. This section is devoted to computing the computational complexity of d considering the queueing behavior of tasks, i.e. tasks being sent to the waiting queue.

In this section, we study the computational complexity of the RDM, where we have only one waiting queue connected to all the existing computing units. Furthermore, we investigate also the computational complexity of the search space d of the system in case the type of queue is a FIFO queue or free access waiting queue. In addition, we estimate the lower bound and the upper bound of the search space d of the system.

6.3.1 Shared FIFO waiting queue in the system

Let's compute the computational complexity of the search space d in case of considering a shared FIFO waiting queue to all computing units and multi-task generators.

In the sequel, we assume that we have a FIFO queue (subtasks case is not considered, handling only task types). Let q be the actual number of tasks in the waiting queue including the new task, where tasks are coming from different task generators g. The first task in the queue means the first one which leaves the queue. We can deploy theoretically the 1st task to one of the K computing units. Similarly, we can do the same with the 2nd, etc. As it is noticeable when we deploy a certain task we have to consider the actual state of the computing units. As it is noticeable when we deploy a certain task we have to consider the actual state of the computing units.

In case the waiting queue is empty, the number of possible scenarios is K. While, in case the waiting queue is not empty, we need to deploy the q tasks in K^q different ways. This K^q is the size of the search space. In this case, one writes $d = K^q$.

Now, let's compute the computational complexity of the search space d in the case of considering a shared FIFO waiting queue connected to all computing units, and multi-tasks, multi-subtasks generators.

The expression in Eq. (4.13) presents the total possible number of deployment scenarios if the FIFO waiting queue contains only one task. Now, if we consider that the waiting queue contains q tasks, then the K different scenarios in K^q have to be replaced by $\prod_{v=1}^{W} \binom{K + M_{gv} - 1}{M_{gv}}$ as we

have earlier discussed in the multi-tasks case. To this end, the value of d in multi-tasks and the multi-subtasks case can be written as

$$d = \left(\prod_{v=1}^{W} \binom{K + M_{gv} - 1}{M_{gv}} \right)^{q}. \tag{6.6}$$

Now, we are in a position to estimate the lower and the upper bounds of the search space d in the case of the multi-tasks, multi-subtasks model, where the type of the waiting queue is FIFO queue. Similarly as before (Chapter 4), one can easily verify that the lower and upper bounds can be expressed respectively as

$$d = \left(\prod_{v=1}^{W} \binom{K + M_{gv} - 1}{M_{gv}} \right)^{q} \leq \left(\frac{K^{M_{gv}}}{\min\limits_{v} M_{gv}} \right)^{W.q}. \tag{6.7}$$

$$d \geq \left(\frac{K^{M_{gv}}}{\max\limits_{v} M_{gv}\,!} \right)^{W.q}. \tag{6.8}$$

6.3.2 Shared free access waiting queue in the system

Let's compute the computational complexity of the search space d in case of considering a shared free access waiting queue connected to all computing units and multi-task generators. In this case, the q tasks in the free access waiting queue can be permuted $q!$ times. Hence, it is straightforward to verify that the value of d becomes $d = q!\ K^{q}$.

Let's compute the computational complexity of the search space d in case of considering a shared free access waiting queue with multi-tasks and multi-subtasks generators.

It is interesting to note that by using $d = q!\ K^{q}$, the value of d in the case of considering a shared free access waiting queue with multi-tasks and multi-subtasks generators can be represented as

$$q!\left(\prod_{v=1}^{W}\binom{K+M_{gv}-1}{M_{gv}}\right)^{q}. \tag{6.12}$$

In a similar way as before, to estimate the lower and the upper bounds of the search space *d* in the case of the multi-tasks, multi-subtasks system, where the type of the waiting queue is free access queue one can easily verify the upper and lower bounds of Eq. (6.12) which are represented respectively as follows

$$d = q!\left(\prod_{v=1}^{W}\binom{K+M_{gv}-1}{M_{gv}}\right)^{q} \leq q^{q}\left(\frac{K^{M_{gv}}}{\min\limits_{v} M_{gv}}\right)^{Wq} \tag{6.13}$$

$$d \geq q\left(\frac{K^{M_{gv}}}{\max\limits_{v} M_{gv}\,!}\right)^{W.q} \tag{6.14}$$

6.3.3 Analytical results

It is clearly shown from Eq. (6.7) _FIFO waiting queue and multi-tasks, multi-subtasks_ that the computational complexity of finding the optimal deployment scenario within the database is polynomial in terms of the number of computing units *K* but exponential in terms of M_{gv}, i.e. if the number of subtask type *W* increases, the computational complexity augments significantly.

On the other hand, in case of free access waiting queue and multi-tasks, multi-subtasks, and using Eq. (6.13), it is noticeable that the computational complexity of selecting the optimal distribution scenario within the database is polynomial in terms of the numbers of computing unit *K* and q but exponential in terms of M_{gv}.

We sum up that the computing power needed to achieve the task deployment in case of waiting queue is beyond the capacities of classical methods. To this end, applying a constrained classical computation method to find the optimum distribution scenario that corresponds to the minimum energy consumption will be time-consuming and hard to solve.

The proposed implemented quantum strategy is relying on the CQRT derived from quantum phase estimation which constitutes a fundamental dynamo of the proposed CQOA, where it allows solving problems with an exponential speed-up.

6.4 Simulation Results

To demonstrate the efficiency of the proposed CQOA, a simulation environment was constructed to compare the performance of the proposed constrained optimized solution and the constrained randomized one.

We built three experiments that are described as follows,

- Case A (normal operation of the RDMS): The system does not reach the overload state of the RDMS, i.e. no queue is considered (see subsection 6.4.1).
- Case B (the overloaded scenarios with full rejection are considered): The new incoming task is rejected if and only if one of the constraints is not met, i.e. either the capacity constraint or delay constraint is not satisfied (see subsection 6.4.2).
- Case C (overloaded scenarios with considering partial rejection): The task is rejected when the delay constraint is not gratified, while it is sent to the waiting queue when there is no free space in the computing units (see subsection 6.4.2).

It is interesting to note that the experiments aim to evaluate and compare the performance of both strategies (The optimized and the randomized one) in terms of

- The overall energy consumption by taking into account the constraint delay.
- The computational complexity.
- The number of rejected tasks.

The simulation environment hosts three computing units with an identical processing speed of 40. Here we consider the situation where we have a _Free access waiting queue and multi-tasks, multi-subtasks_. Also, we consider two task-type generators and three subtask types, and two decision-makers, the first one is dedicated to the constrained optimized strategy, and the second one is devoted to the constrained randomized method.

From a practical point of view, the RDMS contains dramatically more computing units, however, to evaluate the trends and effects of the new quantum strategy. It is worthwhile studying a small-scale system. Figure 6.2 shows the components of the considered RDMS.

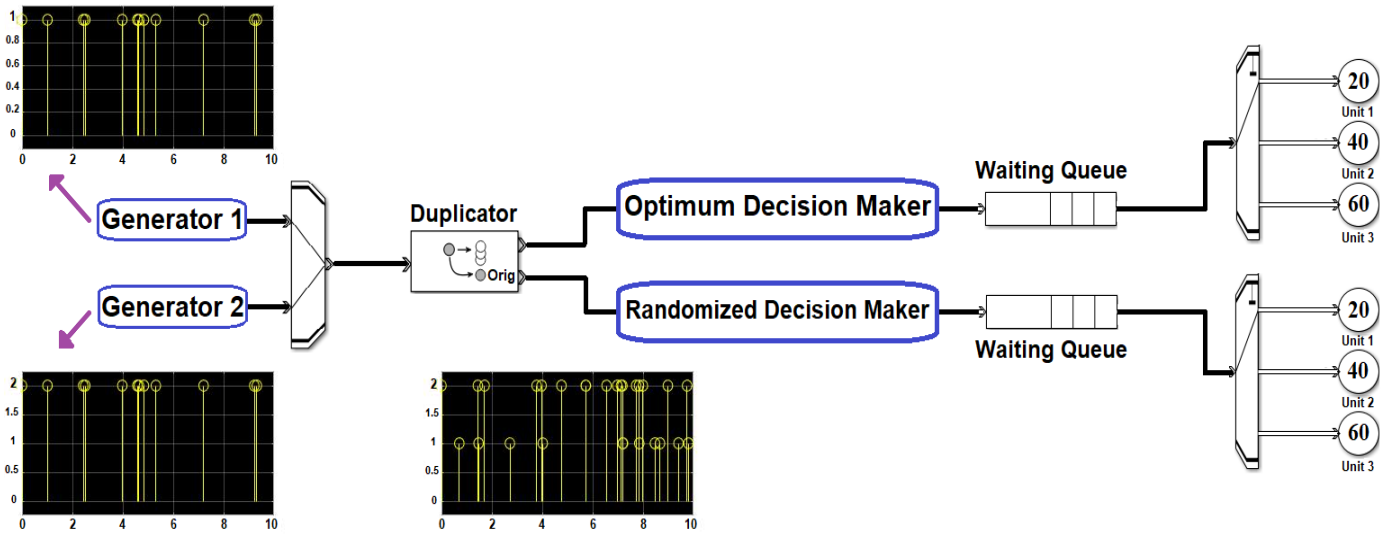


Figure 6.2: Simulation environment

The delay constraint of the first and second task types are respectively 1.4 and 1.2. The timeslot of task generation is 20 seconds. The memory size of the subtask types is presented in Table. 4.1. The number of subtask types in each task type is illustrated in Table. 4.2, while the characteristics of computing units are described in Table. 6.1.

Table 6.1: Characteristics of computing units.

	Computing Unit 1	**Computing Unit 2**	**Computing Unit 3**
c_k	20	40	60
P_k^{init}	20	25	30
ε_k	1	2	3

We repeat the simulation for different intensity distributions (Here we consider exponential intensity distribution). Table. 6.2 presents the number of tasks generated for every intensity distribution. The notation $M\_1 = 0.1$ denotes the mean value of the exponential distribution equals 0.1, and N refers to the number of generated tasks to the corresponding intensity distribution.

Table 6.2: The number of generated tasks in each intensity distribution (length of time generation is 20).

	***M*_1=0.2**	***M*_2=0.1**	***M*_3=0.09**	***M*_4=0.08**	***M*_5=0.05**	***M*_6=0.04**	***M*_7=0.02**	***M*_8=0.01**
N	91	203	226	260	411	517	812	841

6.4.1 Case A

Here we consider that the RDMS is in the normal operation of the system where no queue is considered. For the two intensity distributions M_1 and M_2, we repeated the simulation 10 times, then we calculated the average total energy consumption for each strategy, i.e. the constrained optimized algorithm and the constrained randomized one.

As it is shown in Table. 6.3, the number of rejected tasks is equal to zero when the intensity distribution of the task types corresponding respectively to $M\_1 = 0.1$ and $M\_2 = 0.2$, i.e. it means that the system is not overloaded yet. Furthermore, it is clearly noticeable that the total energy consumption of the constrained randomized strategy significantly higher than the constrained optimized one.

The computational complexity of the constrained randomized method is $O(1)$ or $O(const)$, while in the worst-case, it is $O(d)$ steps. While the computational complexity of the CQOA is $O\left(log_2(T)log_2{}^3(\sqrt{d})\right)$. As a result, the proposed constrained optimized approach makes a substantial reduction in terms of computational complexity and energy usage.

Table 6.3: Comparison results between the constrained optimized strategy (OPT) and the constrained randomized one in Case A, multi-tasks, multi-subtask.

Normal operating Case	ANRT_OPT	ANRT_RAN	RfR_OPT	RfR_RAN	PoRT_OPT	PoRT_RAN	TEC_OPT	TEC_RAN
*M*_1 & *M*_2 *S* = 294 Low Intensity	0	0	_	_	0%	0%	3039.85	3568.68

OPT:	Optimized strategy
RAN:	Randomized strategy
ANRT_OPT:	Average number of rejected tasks OPT
ANRT_RAN:	Average number of rejected tasks RAN
RfR_OPT:	Reason for rejection OPT
RfR_RAN:	Reason for rejection RAN
PoRT_OPT:	Percentage of rejected tasks OPT (%)
PoRT_RAN:	Percentage of rejected tasks RAN (%)
TEC_OPT:	Total Energy Consumption OPT
TEC_RAN:	Total Energy Consumption RAN
S:	Total number of generated tasks for both generators

6.4.2 Case B

Here we consider that the new arrival task is rejected if and only if the capacity constraint or delay constraint is not met.

For each two different intensity distributions, we repeated the simulation 10 times, then we calculated the average number of rejected tasks, the reason for rejection, the percentage of rejected tasks, and the average total energy consumption for each strategy, i.e. the constrained optimized algorithm and the constrained randomized one.

Table. 6.4 shows a comparison between the constrained optimized method and the constrained randomized one in Case B. Note that the notation "DCNM" which refers to the delay constraint is not met, while "CCNM" refers to capacity constraint is not met.

It is interesting to note that when the mean value of the exponential distribution of the arrival times of tasks is smaller (the tasks are generated faster), the number of rejected tasks augments simultaneously. According to Table. 6.4, the columns corresponding to the average number of rejected tasks show that the number of rejected tasks of the constrained optimized strategy is approximately equal to the one of the constrained randomized case.

We can also visualize that when the intensity distribution increases, the total energy consumed by the optimized strategy is lower than that of the energy consumed by the randomized method. For example, when the intensity distribution of the task types corresponding respectively to $M\_5 = 0.5$ and $M\_6 = 0.4$, the total energy consumption of the constrained optimized method equals 5748.2 while the energy consumption of the randomized strategy is 5803.47.

As it is known, in the optimal case, the computational complexity of the constrained randomized method is $O(1)$ or $O(const)$, while in the worst-case, it is $O(d)$. In contrast, the computational complexity of the CQOA is $O\left(log_2(T)log_2{}^3\left(\sqrt{d}\right)\right)$. Consequently, one can sum up that the proposed constrained optimized strategy provides a significant reduction in terms of computational complexity and energy consumption.

Table 6.4: Comparison results between the constrained optimized strategy (OPT) and the constrained randomized one in Case B, multi-tasks, multi-subtask.

Rejection Case	ANRT_OPT	ANRT_RAN	RfR_OPT	RfR_RAN	PoRT_OPT	PoRT_RAN	TEC_OPT	TEC_RAN
M_2 & *M_3* $S = 429$ Medium Intensity	18	17	DCNM	DCNM	4.19%	3.96%	5111.85	5310.7
M_3 & *M_4* $S = 486$ High Intensity	62	58	DCNM	DCNM	12.75	11.93	5277.15	5439.97
M_5 & *M_6* $S = 928$ High+ Intensity	452	454	CCNM=42 DCNM=410	CCNM=68 DCNM=386	48.70	48.90	5748.2	5803.47
M_7 & *M_8* $S = 1653$ High++ Intensity	1197	1199	CCNM=216 DCNM=981	CCNM=195 DCNM=1004	72.41	72.60	5767.12	5783.2

OPT:	Optimized strategy
RAN:	Randomized strategy
ANRT_OPT:	Average number of rejected tasks OPT
ANRT_RAN:	Average number of rejected tasks RAN
RfR_OPT:	Reason for rejection OPT
RfR_RAN:	Reason for rejection RAN
PoRT_OPT:	Percentage of rejected tasks OPT (%)
PoRT_RAN:	Percentage of rejected tasks RAN (%)
TEC_OPT:	Total Energy Consumption OPT
TEC_RAN:	Total Energy Consumption RAN
DCNM:	Delay constraint not met
CCNM:	Capacity constraint not met
S:	Total number of generated tasks for both generators

6.4.3 Case C

In this case, we consider that the task is rejected when the delay constraint is not met and it is sent to the waiting queue when there is no free space in the computing units.

According to what has been described previously in subsection 6.4.1. For each two different intensity distributions, we repeated the simulation 10 times, then we computed the average number of rejected tasks, the reason for rejection, the average total energy consumption, and the average number of tasks that were served from the waiting queue for both algorithms, i.e. the constrained optimized algorithm and the constrained randomized one. Table. 6.5 shows a comparison between the constrained optimized strategy and the constrained randomized one in the case of free access waiting queue and multi-tasks, multi-subtasks.

Table 6.5: Comparison results between the constrained optimized strategy (OPT) and the constrained randomized one in Case C, free access waiting queue, multi-tasks, multi-subtask _.

Waiting Queue Case	R_D_OPT	R_D_RAN	R_C_OPT	R_C_RAN	STQ_OPT	STQ_RAN	ER_OPT	ER_RAN	TEC_OPT	TEC_RAN
M_5 & *M_6* *S* = 928 High + Intensity	403	423.5	55	61.66	7	5.55	415	453	5760.95	5797.83
M_7 & *M_8* *S* = 1653 High ++ Intensity	1013	1070.33	205	193	24	10.33	1194	1199.66	5789.85	5791.19

OPT: Optimized strategy
RAN: Randomized strategy
R_D_OPT: Average number of rejected tasks because of delay constraint OPT
R_D_RAN: Average number of rejected tasks because of delay constraint RAN
R_C_OPT: Average number of rejected Tasks because of capacity OPT
R_C_RAN: Average number of rejected Tasks because of capacity RAN
STQ_OPT: Average number of served tasks of the WAITING QUEUE (OPT)
STQ_RAN: Average number of served tasks of the WAITING QUEUE (RAN)
ER_OPT: At the end of the Simulation, the average number of rejected tasks by the OPT
ER_RAN: At the end of the Simulation, the average number of rejected tasks by the RAN
TEC_OPT: Total Energy Consumption OPT
TEC_RAN: Total Energy Consumption RAN
S: Total number of generated tasks for both generators

As shown in Table. 6.5, when the system is overloaded, the total number of rejected tasks at the end of the simulation of the constrained randomized strategy is slightly higher than the constrained optimized one. While both methods consume approximately the same quantity of energy. Consequently, both strategies have similar performance in terms of energy consumption when the system is overloaded.

It is important to mention that when the waiting queue is not empty and there is a free capacity space in the system, the task is selected from the waiting queue according to the priority selection. In this paper the priority criteria considered is the largest delay, this delay is the sum of the waiting time in the queue and the time required to serve the task.

The computational complexity for queueing scenarios depends on,

- The computational complexity of selecting the optimum distribution scenario from the overall computing units K.

- ❖ The computational complexity of the selection of the most prior task from the free access waiting queue.

The computational complexity of selecting the optimum deployment scenario for the constrained randomized method is $O(d')$ or $O(1)$ and for selecting the task from the waiting queue according to the priority deployment is $O(q)$. In a similar way as before, It is straightforward to verify that the lower overall possible deployment scenarios d'can be written as follows

$$d' = \prod_{v=1}^{W} \binom{K + M_{gv} - 1}{M_{gv}} \geq \left(\frac{K^{M_{gv}}}{\max\limits_{v} M_{gv}\,!} \right)^{W}. \tag{6.15}$$

This proves that the computational complexity of selecting the optimal distribution scenario within the database is polynomial in terms of the numbers of computing unit K but exponential in terms of M_{gv}. Furthermore, the computational complexity of selecting a task within the free access waiting queue is polynomial in terms of q.

On the other hand, the computational complexity of selecting the best optimum deployment among computing units and the selection of the most prior task from the queue of the constrained quantum solution requires only $O\left(log_2(T) {log_2}^3\left(\sqrt{d}\right)\right)$. As previously shown in Eq. (6.13), the computational complexity of selecting the optimal deployment scenario within the database is polynomial in terms of the numbers of computing unit K and q but exponential in terms of M_{gv}.

In compliance with what has been discussed earlier one can observe that the computational complexity of the constrained quantum solution requires significantly fewer steps than the constrained randomized classical one, additionally, applying the constrained randomized classical solution will be time-consuming and hard to solve.

6.5 Conclusion

In this Chapter, the minimization of the overall constrained energy consumption RDMS with a shared free access waiting queue by exploiting the CQOA has been investigated, as well as the implementation and the configuration of the CQOA.

The upper and the lower bounds of the overall possible deployment scenarios (search space) in the case of considering queueing scenarios have been estimated. The analytical results showed that the constrained assignment optimization problem cannot be performed by the classical constrained optimization methods while the proposed CQOA finds the optimum deployment scenario with high exponential speed up thanks to the special quantum function named CQRT embedded in the constrained quantum strategy.

Simulation experiments show that the CQOA provides a high reduction in terms of computational complexity and energy consumption compared with the most popular used constrained randomized strategy.

7 Summary

This research investigates the capabilities of quantum methods in handling task assignment optimization problems. The contributions can be summarized in the following five thesis groups.

The first result group presents a set of quantum optimization algorithms where each one of them can handle a specific type of database structure and optimization problem (Chapter 2). We also emphasize that the investigated quantum solutions provide significant computational complexity reduction and high accuracy compared with the classical existing ones.

The second result group points out the possibility of load balance of resource distribution management (RDM) system with multiple type task generators if the quantum extreme value searching algorithm (QEVSA) could be embedded in the decision assignment module of (Chapter 3). It suggests a prospective trend of quantum computational infrastructure to enable optimizing any type of unconstrained goal function. The proposed implemented quantum strategy can reach the load balance with less computational complexity and high accuracy. Also, simulation experiments were built to validate the efficiency of the quantum strategy.

The third result group extends the RDM to a more complex task assignment problem where multi-tasks, multi-subtasks are considered (Chapter 4). The QEVSA succeeded to conserve the load balance of the incoming tasks. This type of task deployment optimization problem cannot be solved by ordinary computers because it requires high exponential processing speed up. Thus, the quantum solution can solve the distribution optimization problem with $O\left(log_2(T)log_2{}^3\left(\sqrt{d}\right)\right)$, where T is the maximum number of steps needed to run the logarithmic search embedded in QEVSA and d is the size of all possible deployment scenarios. Finally, extensive simulations were constructed to prove the efficiency of QEVSA.

The fourth result group points out the capability of the constrained quantum optimization algorithm (CQOA) in saving the total constrained energy consumption of the RDM with multi-task, multi-subtasks (Chapter 5). The proposed configuration of the stochastic parameters of CQOA must be calculated carefully. Also, an analytical demonstration proved that the

computational complexity of finding the optimal deployment scenario within the database is polynomial in terms of the numbers of computing units but exponential in terms of the number of subtasks. Thus, the proposed constrained quantum computing power is industriously needed. Finally, simulations proved the efficiency of the suggested CQOA.

The fifth result group investigates the minimization of the overall constrained energy consumption RDM with a shared free access waiting queue by exploiting the CQOA (Chapter 6). The lower bound of the size of the total number of possible distribution scenarios is polynomial in terms of the number of computing units and number of tasks waiting in the queue but exponential in terms of the number of subtasks. Therefore, performing such a classical constrained task assignment method seems impossible. Finally, simulations proved the efficiency of the suggested CQOA, three main results were obtained in case of considering queueing scenarios,

- The constrained randomized algorithm and the constrained optimized method consume approximately the same quantity of energy.
- The total number of rejected tasks at the end of the simulation of the constrained randomized strategy is slightly higher than the constrained optimized one
- The computational complexity of the constrained randomized is $O(d'q)$, where q is the number of tasks in the waiting queue and computational complexity of CQOA is $O\left(log_2(T){log_2}^3(\sqrt{d})\right)$.

In the future work, we would expand the CQOA's functionalities to find the optimum solutions of a constrained multivariable objective function. Following that, the RDM paradigm would be improved with a more challenging task assignment problem in which each computing unit is linked to a waiting queue. Hence, in the latest RDM update, the current expanded version of the CQOA will be used.

Machine-learning, network telecommunications, physical structure simulation, and other problems are concerned with minimizing the constrained multivariate objective function, the contribution of quantum optimization algorithms in these problems may lead to ensuring a high exponential speed and a short execution time.

Appendix

Considering only one task generator, all the tasks have the same memory requirement, moreover, we assume that there is a large number of computing units denoted by *K*, the computing units have different theoretical capacities.

The relative load denoted by b_i is expressed in (1), where x_i is the used capacity of the k^{th} unit and s_k is the theoretical capacity of the k^{th} unit. The average relative load denoted by $\bar{b}$ is given by the formula (2), b_k and x_k are defined as follows,

$$b_k = \frac{x_k}{s_k}. \tag{1}$$

The relative average load is defined such that,

$$\bar{b} = \frac{1}{K} \sum_{i=1}^{K} b_k \tag{2}$$

The suitable α is obtained by using the workload states of two different scenarios, we assume that m is the memory requirement of the new task. the i^{th} computing unit receiving the new task has the relative load $b_{k=i}^{i}$, expressed in (3), the remaining computing units have the relative load $b_{k\neq i}$, expressed in (4).

$$b_{k=i}^{i} = b_i + \frac{m}{s_i} \tag{3}$$

And,

$$b_{k\neq i} = b_k. \tag{4}$$

The variance of the relative load of the i^{th} unit receiving the new task is expressed as follows,

$$\sigma_i^2 = \frac{1}{K} \sum_{k\neq i}^{c} \left(\bar{b_\iota} - b_k\right)^2 + \frac{1}{K}\left(\bar{b_\iota} - b_i + \frac{m}{s_i}\right)^2. \tag{5}$$

We know that $\frac{m}{s_i} < b_k$, and if we assumed that K is a large number,$K \gg 1$, we get the following expression,

$$\sigma_i^2 - \sigma_j^2 \cong \frac{1}{K}\left[\left(\frac{m}{s_i}\right)^2 - \left(\frac{m}{s_j}\right)^2 + 2\left(\frac{m}{s_i}b_i - \frac{m}{s_j}b_j\right)\right] \quad (6)$$

To reach the aforementioned goal, it is necessary to apply some simplifications, so, we substitute b_i by $\frac{N_i .m}{s_i}$ and b_j by $\frac{N_j .m}{s_j}$, where N_i and N_j are the number of tasks respectively in the i^{th} unit and the j^{th} unit. We obtain the following expression,

$$\sigma_i^2 - \sigma_j^2 \cong \frac{1}{K}\left[\left(\frac{m}{s_i}\right)^2 (1 + 2N_i) - \left(\frac{m}{s_j}\right)^2 (1 + 2N_j)\right]. \quad (7)$$

Own Publications

Journal Papers

[J1]: S. El Gaily, S. Imre. EVALUATION OF RESOURCE OPTIMIZATION BASED ON QUANTUM SEARCH. Hungarian Journal of industry and chemistry. 2019. DOI: 10.33927/hjic-2019-03.

[J2]: S. El Gaily, S. Imre, "Quantum Optimization of Resource Distribution Management for Multi-Task, Multi-Subtasks", Infocommunications Journal, Vol. XI, No 4, December 2019, pp. 47-53.

[J3]: S. El Gaily, S. Imre, " Constrained Quantum Optimization for Resource Distribution Management, *Int. J. Adv. Comput. Sci. Appl*, 2021.

[J4]: S. El Gaily, S. Imre, " Implementation of a Constrained Quantum Optimization Method in Resource Distribution Management with Considering Queueing Scenarios ", *Int. J. Communication Networks and Distributed Systems*, 2021.

Conference Papers

[C1] S. El Gaily:" RESOURCE OPTIMIZATION BASED ON QUANTUM SEARCHING", 15th International Ph.D. Workshop on Systems and Control (2018) pp. 1-3.

[C2] S. El Gaily: "Performance evaluation of quantum-based resource management" The International Conference on Quantum Computing (ICoCQ2019), (2018) pp. 1-5.

[C3] S. El Gaily, S. Imre: Quantum Optimization in Large Resource Management Systems - IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2-5 July 2019, Cannes, France, DOI: 10.1109/SPAWC.2019.8815470.

[C4] S. El Gaily, S. Imre: Derivation of Parameters of Quantum optimization in Resource Distribution Management - 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), 1-3 July 2019, Budapest, Hungary, DOI: 10.1109/TSP.2019.8769092.

[C5] S. El Gaily, S. Imre: “Quantum Resource Distribution Management in Multi-task environment”, 14th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS).

[C6] S. El Gaily, S. Imre: “Quantum Computing-based Onboard Hardware Management for Spacecrafts” ITU WORLD 2019. Conference on space communication, space activities and risks of the IT sector for V4 countries”.

[C7] S. El Gaily, S. Imre: “Extending the functionalities of the quantum extreme value searching algorithm to a constrained quantum searching algorithm”, The Sixth International Conference for Young Quantum Information Scientists (YQIS 6 or YQIS 2021). 12-16 April 2021, USA.

[C8] S. El Gaily, S. Imre: “Constrained Quantum Optimization Strategy”, The Bristol Quantum Information Technologies Workshop. 26-28 April. England.

[C9] S. El Gaily, S. Imre: “Constrained Quantum Optimization Algorithm”, 20th International Symposium INFOTECH-JAHORINA, 17-19 March 2021, Jahorina, East Sarajevo.

[C10] S. El Gaily, S. Imre: “Quantum Computing based Efficient Optimization”, CTP Quantum Information Days, 22-24 February 2021, Poland.

Bibliography

[1] Abbasov, B: Cloud computing: State of the art research issues, IEEE 8th International Conference on Application of Information and Communication Technologies (AICT). 15-17 Oct. 2014.

[2] Zhang, Q., Cheng L., and Routaba R. (2010) Cloud Computing: State-of-the-art and Research Challenges, *Journal of Internet Services and Applications* 1(1):7-18.

[3] Mohiuddin A.et al. (2012) An Advanced Survey on Cloud Computing and State-of-the-art Research Issues. *IJCSI International Journal of Computer Science Issues*, Vol. 9, Issue 1, No 1, January 2012.

[4] Shameem, P M., Johnson, N., Shaji, R.S. and Arun, E. An effective resource management in cloud computing, *Int. J. Communication Networks and Distributed Systems*, Vol. 19, No. 4.

[5] Srinivasa K G. (2017) Utility maximisation-based game theoretic approach for resource allocation in clouds, *Int. J. Communication Networks and Distributed Systems*, Vol. 18, No. 3/4.

[6] Jain, A., and Kumar, R. (2017) Hybrid load balancing approach for cloud environment, *Int. J. Communication Networks and Distributed Systems*, Vol. 18, No. 3/4.

[7] Kaddouri, A., Guezouri, M., and Mbarek, N. (2018) A new proposed cloud computing based architecture for space ground data systems, *Int. J. Communication Networks and Distributed Systems*, Vol. 20, No. 2.

[8] Erl T, Puttini R, and Mahloud Z. Cloud Computing : Concepts, Technology & Architecture, 23 May 2013.

[9] Nielsen, M A. and Chuang, I L. Quantum computation and quantum information. Cambridge University Press. (2000)

[10] S. Imre, F. Balázs: Quantum Computing and Communications – An Engineering Approach, Published by John Wiley and Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2005, ISBN 0-470-86902-X, 283 pages. DOI: 10.1002/9780470869048.

[11] Shor P W. 1995 Polynomial-Time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing. 26 5.

[12] Harrow A, Hassidim A, and Lloyd S. 2009 Quantum algorithm for linear systems of equations. Phys. Rev. Lett. 103

[13] Grover L K. 1997 Quantum Computers Can Search Arbitrarily Large Databases by a Single Query, PhysRevLet 79 23.

[14] Grover L K. 1996 Fast quantum mechanical algorithm for database search, STOC '96 Proc. of the Twenty-eighth annual ACM symposium on Theory of Computing.

[15] Srinivasa K G. (2017) Utility maximisation-based game theoretic approach for resource allocation in clouds, *Int. J. Communication Networks and Distributed Systems*, Vol. 18, No. 3/4.

[16] Jain, A., and Kumar, R. (2017) Hybrid load balancing approach for cloud environment, *Int. J. Communication Networks and Distributed Systems*, Vol. 18, No. 3/4.

[17] Kaddouri, A., Guezouri, M., and Mbarek, N. (2018) A new proposed cloud computing based architecture for space ground data systems, *Int. J. Communication Networks and Distributed Systems*, Vol. 20, No. 2.

[18] Farhi E and Goldstone J. 2014 A Quantum approximate optimization algorithm.

[19] Suzette S J, Ciaran R A, William B, and Ojas D P. 2019 Quantum approximate optimization algorithm (QAOA) on constrained optimization problems, 21st Annual Southwest Quantum Information and Technology Workshop.

[20] Hadfield S, Wang Z, O'Gorman B, and Al. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms 12 2. (2019)

[21] Lin, M-H., Tsai, J-F., and Yu, C-S. (2012) A Review of Deterministic Optimization Methods in Engineering and Management, Mathematical Problems in Engineering.

[22] Teok, C K., Wibowo, A., and Ngadiman M S. Review of state of the art for metaheuristic techniques in Academic Scheduling Problems. *Artificial Intelligence Review* 44. (2013)

[23] J.D. Ullman, *NP*-complete scheduling problems, *J Comput Syst Sci*, 10 (3), 1975

[24] P. Arroba et Al, Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy-efficient cloud data centers, *Concurrency and Computation Practice and Experience* 29(10).

[25] L. Zhao, B. Liu, and J. Ma, Virtual machine allocation method for cloud computing based on multi-objective evolutionary algorithm, *Int. J. of Information and Communication Technology (IJICT)*, Vol. 18, No. 2, 2021.

[26] S. Pang et Al, Ant colony optimization algorithm to dynamic energy management in cloud data center, *Mathematical Problems in Engineering*, 2017.

[27] F. U. Zambuk et Al, Efficient Task Scheduling in Cloud Computing using Multi-objective Hybrid Ant Colony Optimization Algorithm for Energy Efficiency, *Int. J. of Advanced Computer Science and Applications*, Vol. 12, No. 3, 2021.

[28] R. Rani and R. Garg, Energy efficient task scheduling using adaptive PSO for cloud computing, Int. J. of Reasoning-based Intelligent Systems, Vol. 13, No. 2, 2021.

[29] P. Azad, N. J. Navimipour, and M. Hosseinzadeh. A fuzzy-based method for task scheduling in the cloud environments using inverted ant colony optimization algorithm, *Int. J. Bio-Inspired Computation*, Vol. 14, No. 2, 2019

[30] Sahraei, S H., Kashani, M M R., Rezazadeh, J., and Farahbakhsh, R. (2019) Efficient job scheduling in cloud computing based on genetic algorithm, *Int. J. Communication Networks and Distributed Systems*, Vol. 22, No. 4.

[31] S. Yang et Al, Energy-saving algorithm for data centre network based on genetic algorithm, *Int. J. of Innovative Computing and Applications (IJICA)*, Vol. 11, No. 2/3, 2020.

[32] Hima Bindu G.B., Ramani, K., and Shoba Bindu, C. Energy aware multi objective genetic algorithm for task scheduling in cloud computing, *Int. J. Internet Protocol Technology*, Vol. 11, No. 4. 2018.

[33] S. Srichandan, T. A. Kumar, and S. Bibhudatta, Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm, *Future Computing and Informatics Journal*, Vol. 3, No. 2, 2018.

[34] S. K. Mishra, B. Sahoo, and P. P. Parida, Load balancing in cloud computing: A big picture, *Journal of King Saud University - Computer and Information Sciences*, Vol. 32, No. 2, 2020.

[35] A. Pradhan, S. K. Bisoy, P. K. Mallick, Load Balancing in Cloud Computing: Survey, *Innovation in Electrical Power Engineering, Communication, and Computing Technology*, 2020.

[36] Balusamy, B., Karthikeyan, K., and Sangaiah A K. Ant colony-based load balancing and fault recovery for cloud computing environment, *Int. J. Advanced Intelligence Paradigms*, Vol. 9. No. 2/3. 2017.

[37] Kumar, R., and Prashar, T. A bio-inspired hybrid algorithm for effective load balancing in cloud computing, *Int. J. Cloud Computing*, Vol. 5. No. 3. 2016.

[38] Rajguru, A A., and Apte, S S. QoS enhanced distributed load balancing and task scheduling framework for wireless networks using hybrid optimisation algorithm, *Int. J. Communication Networks and Distributed Systems*, Vol. 21, No. 2. 2018

[39] Meenakshi Priya, A., and Kannigy Devi, R. (2019) Multi-objective optimisation techniques for virtual machine migration-based load balancing in cloud data centre, *Int. J. Cloud Computing, Vol*. 8. No. 3

[40] S. Imre: „Extreme Value Searching in Unsorted Databases Based on Quantum Computing", International Journal of Quantum Information, Published by World Scientific Publishing Company, ISSN 0219-7499, Vol. 3. No. 1. 2005, pp. 171-176. DOI: 10.1142/S0219749905000700.

[41] S. Imre, "Quantum Existence Testing and Its Application for Finding Extreme Values in Unsorted Databases" IEEE Transactions on Computers, Vol: 56, Issue: 5, May 2007, DOI: 10.1109/TC.2007.1032.

[42] Knuth, Donald (1998). Sorting and searching. The Art of Computer Programming. 3 (2nd ed.). Reading, MA: Addison-Wesley Professional. ISBN 978-0-201-89685-5. DOI: 10.1090/s0002-9904-1973-13173-8.

[43] N. Ariffin, M. Zin, S. Norul, H. Sheikh, N. Faridatul, A. Zainal: A Comparison of Exhaustive, Heuristic and Genetic Algorithm for Travelling Salesman Problem in PROLOG, International Journal on Advanced Science Engineering Information Technology, 2012.vol 2 (2012) No.6 ISSN:2088-5334.

[44] J.N. Hooker: Unifying Local and Exhaustive Search, Carnegie Mellon University, September 2005. DOI: 10.1111/itor.12020