Article

Development of Evolutionary Systems Based on Quantum Petri Nets

Tiberiu Stefan Letia, Elenita Maria Durla-Pasca, Dahlia Al-Janabi and Octavian Petru Cuibus

Special Issue
Information Theory Applied in Scientific Computing

Edited by
Prof. Dr. Tiberiu Letia and Prof. Dr. Alexandru Onea

https://doi.org/10.3390/math10234404

Article

Development of Evolutionary Systems Based on Quantum Petri Nets

Tiberiu Stefan Letia *, Elenita Maria Durla-Pasca, Dahlia Al-Janabi and Octavian Petru Cuibus

Department of Automation, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania
* Correspondence: tiberiu.letia@aut.utcluj.ro

Abstract: Evolutionary systems (ES) include software applications that solve problems using heuristic methods instead of the deterministic ones. The classical computing used for ES development involves random methods to improve different kinds of genomes. The mappings of these genomes lead to individuals that correspond to the searched solutions. The individual evaluations by simulations serve for the improvement of their genotypes. Quantum computations, unlike the classical computations, can describe and simulate a large set of individuals simultaneously. This feature is used to diminish the time for finding the solutions. Quantum Petri Nets (QPNs) can model dynamical systems with probabilistic features that make them appropriate for the development of ES. Some examples of ES applications using the QPNs are given to show the benefits of the current approach. The current research solves quantum evolutionary problems using quantum genetic algorithms conceived and improved based on QPN. They were tested on a dynamic system using a Quantum Discrete Controlled Walker (QDCW).

Keywords: quantum computing; genetic algorithms; Petri nets; quantum Petri nets; software development, analysis and verification

MSC: 68Q12

check for updates

Citation: Letia, T.S.; Durla-Pasca, E.M.; Al-Janabi, D.; Cuibus, O.P. Development of Evolutionary Systems Based on Quantum Petri Nets. *Mathematics* **2022**, *10*, 4404. https://doi.org/10.3390/math10234404

Academic Editor: Jan Sładkowski

Received: 17 August 2022
Accepted: 16 November 2022
Published: 22 November 2022

1. Introduction

The current research concerns applications, such as urban vehicle traffic, airplane traffic, weather, markets, electric power networks, fluids networks, telecommunications and data transmission, etc., that have stochastic behaviors. There are some entities that have to be controlled or managed to behave according to the required specifications.

Quantum computers have different capabilities compared to classical computers and their use should cover the domains where these differences would lead to significant benefits. The authors of [1] review the quantum algorithms that have been discovered and reveal their features that outperform classical algorithms.

According to [2], tools are needed to create and debug quantum computers and their programs. These tools can help to elucidate hidden issues and drive towards design with the best chance for overall success.

The current approach developed methods that can sustain the heuristic search of evaluated solutions that exceed specified thresholds. The previously conceived ES methods are modified according to the features provided by the quantum algorithms.

The general objective in simulating a quantum system is to determine its structure or behavior, given knowledge of its components and the environment in which it exists.

In [2] the quantum computers are partitioned in three categories:

- Analogue quantum computers that directly manipulate the interactions between qubits without breaking the operations in logic gates.
- Digital noisy intermediate-scale quantum computers that use primitive gate operations on physical qubits.

- Fully error-corrected quantum computers that enable noisy physical qubits to emulate stable logical qubits.

The QPNs (Quantum Petri Nets) can be used to model the quantum software applications, to analyze their structure and to verify their behaviors [3]. After the simulations of the QPN models, the implementation of the quantum applications is expected to be achieved (compiled) without difficulties.

This article shows the links between quantum algorithms and their corresponding QPN models. These can sustain the quantum processor configuration and quantum program conception, verification and debugging.

The two directions for solving the problems concern the contraption of methods that use quantum computation methods implemented on classical computers and to create new concepts that allow the finding of the ES solutions using the quantum computers.

1.1. Current Research State of the Field

1.1.1. Quantum Algorithms

Quantum Computing (QC) is recommended to be used in applications where it outperforms the classical computation. Some reviews ([4,5]) present as main QC directions: breaking cryptosystems [6], unstructured search problems, finding accurate approximate solutions to optimization problems, finding the minimum of an unsorted list of integers, determining a graph connectivity, solving linear equations, quantum annealing, etc.

The search in an unstructured database is obtained in [7] using a function $F(x), x \in X = \{0, 1, \ldots, (n-1)\}$, with $F(x) = 0$, for $\forall x \in X$, $x \neq j$ where $F(j) = 1$. The problem consists of finding j. The exponential speedup of the search compared to the classical computation is an undoubted benefit.

Shor's algorithm solves the factorization problem [8]. It is given $F(x) = a^x mod_N$ with $a \in \mathbb{R}, 1 < a < N$. The request is the finding of the smallest integer r such that $a^r mod_N = 1$. Again, the algorithm's exponential search determines its use in many applications, such as the use of QC in cryptography [6,9] .

The inverse transform of a matrix introduced in [10] is used for solving linear equations of the form $A \cdot x = b$, where A is a Hermitean $N \times N$ matrix and b a given unit vector. QC solves such problems that appear in many practical applications with exponential speedup.

Hybrid methods involving quantum and classical computers can be used as Variational Quantum Eigensolver [11,12]. The eigenvalues and eigenvectors of a Hamiltonian are used to transform a system iteratively to a desired specified one.

Quantum annealing is an optimization method for combinatorial problems where the superposition is used for avoiding the focalization in local minimums [13].

Quantum walk opens ways for algorithm construction that solves search problems on a graph [14].

According to [15], the classical random walk concept has been used as a computational framework for designing classical algorithms for complex problems, while quantum variants provide a speed-up in computational power for various algorithms with distinct elements in spatial search or graph connectivity. A quantum walker following a unitary operation for evolution is named a unitary quantum walker, while one that does not meet this condition is a non-unitary quantum walker. The non-unitary evolution of a quantum system is implemented/defined by adding some qubits to the quantum system followed by a unitary transformation. The result is obtained by discarding the extra qubits. Both kinds of walker have benefits in practical applications.

An example is the optimization of a search based on random walks that require the reduction of the number of necessary repetitions [16,17].

Some quantum walker approaches tackle the Quantum Discrete Random Walkers (QDRWs) in an infinite space or finite space [18]. Other researchers studied the quantum continuous random walkers (in an infinite space) [19,20]. They are widely used in practical applications [5]. An application of quantum discrete walker ranking in grid nodes is given in [21].

The reference [22] contains some variants of Quantum Discrete Walker (QRW) approaches and different variants of generalized coin tossing. In [23], two entangled coins were used to control the walker's move in a one-dimension space.

One goal in walker problems is the reducing of the hitting time and the diminishing of mixing time [24].

Quantum walk can be used for search of element distinctness (e.g., find two equal elements) [25].

The comprehensive review [26] divides the quantum walker approaches as Schrodinger and combinatorial. To be noted is the decoherence definition as a physical phenomenon that typically arises from the interaction of quantum systems and their environment. This interaction can be used for quantum process control as it will be seen further for controlling the walker moves.

Quantum Approximate Optimization Algorithm (QAOA) uses the mapping of the objective function to Hamiltonian that brings the problem into Hilbert space [27–29]. The expectation value of the Hamiltonian is improved by using quantum mechanical techniques in the Hilbert space. QAOA can be extended by adding constraints. A relevant problem is the so called MaxCut where a graph $G = (V, E)$ with V vertices and E edges has to be cut in such a manner that it maximizes the number of edges crossing the cut.

In conclusion, the main quantum walk research directions and their practical applications focus on *the hitting times*, *the quantum amplification control* and *the marked element detection*.

1.1.2. Evolutionary Systems

Evolutionary Algorithms (EAs) are applied to problems where pure stochastic algorithms fail or find it difficult to solve due to the high number of dimensions or function complexities [30]. Their main utilizations cover the domains of variable optimization, new structural design and improvement. EAs include: evolutionary strategies, genetic algorithms, genetic programming, genetic improvement, grammatical evolution, linear genetic programming, Cartesian genetic programming, differential evolution and gene expression evolution.

The evolutionary systems implemented on classical computers are often used for solving problems concerning:

- Combinatorial requirements,
- Optimization of system parameters,
- Dynamic behavior optimizations,
- Algorithm synthesis such as for controlling plants or for reacting to events produced by their environments.

Related to the use of QC in EA, some questions arise: What are the benefits of using QC in EA? What are the modifications required for application of QC in EA? Where and how can Quantum Evolutionary Algorithms (QEAs) be applied in practical applications? References [31,32] offer some answers to these.

The use of QC in EAs led to three main development directions: quantum inspired evolutionary algorithms ([33–46]), quantum evolutionary algorithms ([47–53]) and hybrid quantum-classic evolutionary algorithms ([54–56]).

Successfully detailed implementations at the Quantum Logic Circuit (QLC) level are developed and deeply analyzed in [57,58].

Some QEA used animal behaviors for searching the solutions [39,59,60]. Remarkable are the characteristics of the systems that can be approached: they could have various complexities, including non-linear, non-convex, multi-modality, non-differentiable functions and large-scale dimensionality.

Quantum control based on machine learning in an open quantum system is another direction of QC application development [61].

Closest to the view point of quantum state superposition, created by QC, is the Particle Swarm Optimization (PSO) method [62].

Analyzing the QEAs, it can be concluded that the genotypes are coded based on the quantum superposition fulfilling the Hilbert condition. The genotypes are initialized to meet some diversity requirements and covering the search domain. The individual (i.e., genotypes) are improved using unitary transformation matrix. The individual improvements can be based on a single (usually the best) individual or combining features from two individuals (i.e., crossover). Some evolution methods need genotype (or individual) repair mechanisms. There are improvements consisting of the search step adaptations, usually related to the generation iterations. It is difficult to prove the benefits of using the quantum vector representation instead of classical ones. There would be clear benefits if it can be proved that the number of searches and evaluations is smaller in the case of quantum description. The comparisons based on experiments of different QEA methods for solving particular problems can be supposed to be biased to particular characteristics.

The approaches of discrete oriented quantum random walker problems were used to show the benefits of the QPN models.

As our main contributions should be mentioned: the manner of QPN definition, their properties, analysis and verification methods. The Quantum Genetic Algorithms (QGAs) were adapted using the QPNs. They were experimented and tested on a dynamic system. A new kind of quantum discrete random walker model, based on QPN, was used for testing the QGAs. A new quantum fitness method was conceived for this purpose.

2. Materials and Methods

2.1. Quantum State versus Classical State

The classical bits take values in the domain $\{0, 1\}$. They are used to compose the digital information. A number of r bits can cover a domain of 2^r values.

The quantum replacement of the classical bit is the qubit denoted here $|b_k\rangle$: $|b_k\rangle = \alpha_k|0\rangle + \beta_k|1\rangle$ where the complex numbers α_k and β_k satisfy the relation $|\alpha_k|^2 + |\beta_k|^2 = 1$ and $|0\rangle, |1\rangle$ represent their quantum state.

The qubit can be represented on Bloch sphere (Figure 1). It can be described using the angles by the relation:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}e^{j\phi}|1\rangle, \tag{1}$$

with $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$.

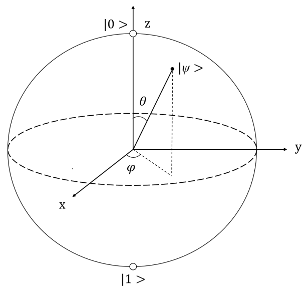

Figure 1. Qubit representation on Bloch sphere.

Joining a number of g qubits leads to a quantum register modeled by qubit vectors of the form:

$$V = [(\alpha_0, \beta_0), (\alpha_1, \beta_1), \ldots, (\alpha_{g-1}, \beta_{g-1})]. \tag{2}$$

The initialization and measurement (i.e., collapse) in quantum computers are performed on qubits. These refer to the V vectors.

The qubits (similar to the bits) can be coded at the computational level by state vectors. The content of a register composed of g (length) qubits can store information using 2^g state vectors from the set $Q = \{|q_0\rangle, |q_1\rangle, \ldots, |q_{r-1}\rangle\}$ with $r = 2^g$. These vectors compose a system of axes used for representing the data in quantum computation.

A quantum register content can be described by the superposed state vector $|\psi\rangle$ in a Hilbert space ($\mathcal{H}$) at the computational base:

$$|\psi\rangle = \sum_{k=0}^{r-1} c_k |q_k\rangle, \tag{3}$$

with $c_k; i = 0, \ldots, r-1$ complex numbers from the set $\mathbb{C}$ fulfilling the relation: $\sum_{k=0}^{r-1} |c_k|^2 = 1$. These coefficients are named amplitudes and they represent the probability distribution of the quantum state. The value $|c_k|^2$ provides the probability of the quantum process to be in the quantum state $|q_k\rangle$. This conception of storing the quantum information based on the quantum state amplitudes (i.e., the coefficients c_i) can sustain the opposite behavior manners named *quantum interference*.

In conclusion, the Hilbert space $\mathcal{H}_q$ has the base $|q_i\rangle : i = 0, 1, \ldots, r-1$ spanning in $\mathcal{H}_q = span\{|q_i\rangle : i = 0, 1, \ldots, r-1\}$.

Let $|\psi_1\rangle = \sum_{i=0}^{r-1} c_i^1 |q_i\rangle$ and $|\psi_2\rangle$ be two vectors in Hilbert space. Then:

$$|\psi_2\rangle = A|\psi_1\rangle = A \sum_{i=0}^{r-1} c_i^1 |q_i\rangle = \sum_{i=0}^{r-1} c_i^1 A |q_i\rangle. \tag{4}$$

A is a matrix that generates transformation of superposed quantum state. $AA^\dagger = A^\dagger A = I$, where $A^\dagger$ is the conjugate transpose of A. A is a so-called unitary matrix. This property of A grants that the transformation $C_2^T = A \cdot C_1^T$ with $C_i = [c_0^i, c_1^i, \ldots, c_r^i]$, $(i = 1, 2)$ is a valid one.

While $|\psi\rangle$ requires 2^g complex coefficients, V involves only $2g$ complex coefficients. These coefficients are linked by the relations:

$$\begin{array}{c} |q_0\rangle = |00\ldots00\rangle; c_0 = \alpha_0\alpha_1\ldots\alpha_{g-1}, \\ |q_1\rangle = |00\ldots01\rangle; c_1 = \beta_0\alpha_1\ldots\alpha_{g-1}, \\ \ldots\ldots \\ |q_{r-1}\rangle = |11\ldots11\rangle; c_{r-1} = \beta_0\beta_1\ldots\beta_{g-1}. \end{array} \tag{5}$$

The conversion of V to Ψ can always be performed, while the reversal (i.e., from Ψ to V) can be done only for some particular (pure) states.

2.2. Quantum Systems versus Classic Dynamic Systems

Let $|\psi_x\rangle = \sum_{k=0}^{r-1} c_k^x |q_k\rangle$ be a quantum vector and $\mathbf{x} = [x_0, x_1, \ldots, x_{r-1}]^T$ a vector with real number elements in the classic system theory. The relation between the quantum vector and the classic vector is set in the current approach by $|c_k^x|^2 = x_k$ for all $k = 0, 1, \ldots, r-1$. As it can be seen, the values x_k $(k = 0, 1, \ldots, r-1)$ correspond to the system probabilities to be in the quantum states $|q_k\rangle$ $(k = 0, 1, \ldots, r-1)$.

Denoting with $X(\tau) = [c_0^x, c_1^x, \ldots, c_7^x]^T$ the vector provided by $|\psi_x\rangle$ at a discrete time τ, a classic dynamic system can be described by $X(\tau+1) = U \cdot X(\tau)$. If U is a unitary matrix, the walker's position, provided by X, remains in Hilbert space (i.e., on the circle domain).

If $|\psi_o\rangle = \sum_{k=0}^{1} c_k^o |q_k\rangle$ is a Hilbert vector, then $|\psi_x^o\rangle = |\psi_o\rangle \otimes |\psi_x\rangle$ is a Hilbert vector as well that extends the position with the entity orientation.

A quantum system $|\psi_x(\tau+1)\rangle = A \cdot |\psi_x(\tau)\rangle$ can model an un-oriented walker, while $\psi_x^o(\tau+1) = A^o \cdot \psi_x^o(\tau)$ can model an oriented one. The un-oriented walker has the state $|\psi_x\rangle$ equivalent to X, while the oriented walker has the state $|\psi_x^o\rangle$ equivalent to (c_0^o, c_1^o, X). The matrices A and A^o have to be unitary for the walker system to remain in the Hilbert space.

2.3. Quantum Computation

Unlike the classical computation that uses logical gates, the processing of quantum information is performed by quantum processors composed of quantum logical gates. A quantum logic gate acting on a qubit is a 2 x 2 matrix. For example, the Hadamard (or diffusion) matrix:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{6}$$

acting on the qubit $|b\rangle = \alpha|0\rangle + \beta|1\rangle$, provides $|b'\rangle = \frac{1}{\sqrt{2}}(\alpha+\beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)|1\rangle$ that verifies the quantum coefficients (i.e., Hilbert) condition.

A single quantum logic gate acts on a qubit, while multiple quantum gates, composing a *quantum logic circuit* act on a register (vector) of qubits, performing an operation of the form $|\psi'\rangle = A|\psi\rangle$. Any such transformation of information has to fulfill the quantum coefficient condition that is met if the corresponding matrix A is unitary.

According to [63], a single qubit quantum gate can be applied to a register of r qubits on a single qubit of a quantum vector. If the qubit is in the k position, the full quantum gate matrix $H$is described by:

$$A_k = \bigotimes_{j=0}^{r-1} \begin{cases} H, & \text{if } j = \text{k} \\ I, & \text{otherwise} \end{cases}, \tag{7}$$

where I is the identity matrix.

For $g = 3$ qubits ($|b_0\rangle, |b_1\rangle, |b_2\rangle$), the application of H on the middle qubit leads to the matrix $A_1 = I \bigotimes H \bigotimes I$ which is a matrix of 8 x 8 size.

Figure 2 represents a quantum logic circuit that acts on a vector, composed of 3 qubits, performing an increment operation detailed by the AND and exclusive OR operations.

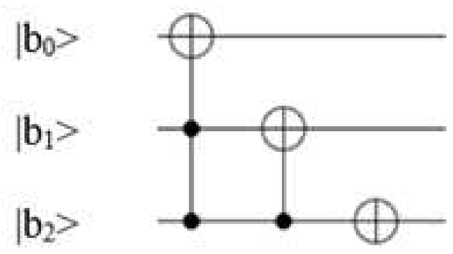

Figure 2. Representation of a quantum vector and quantum operations.

2.4. Quantum Petri Nets

A quantum program involves modifications of the information stored in a set of quantum registers by the quantum gate circuits that link them. This can be modeled by *Quantum Petri Nets* (QPNs) [3]. Figure 3 shows an example of a QPN model that is used for their definition.

Other QPN formal definitions and modeling methods can be found in [64,65]. These show the connections between Petri nets, quantum physics and category theory. The monoidal categories are used for constructing a net theory. In [65], the definitions of QPNs were enriched with formal construction of the reachability graphs.

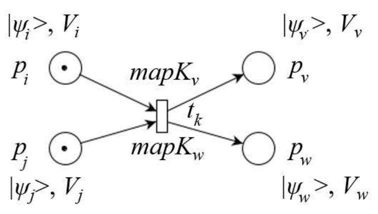

Figure 3. Quantum Petri net model.

Figure 4 shows a QPN modeling a quantum logic circuit that sequentially applies the mappings (i.e., gate operations) A_1, A_2, A_3 to the qubits $|b_0\rangle$, $|b_1\rangle$ and $|b_2\rangle$, respectively, performing the operations $|\psi_4\rangle = A_3 \cdot |\psi_3\rangle = A_3 \cdot A_2 \cdot |\psi_2\rangle = A_3 \cdot A_2 \cdot A_1 \cdot |\psi_1\rangle$.

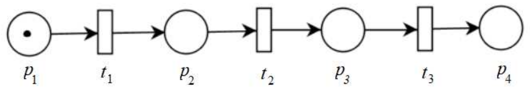

Figure 4. Quantum Petri net model of Figure 2 representation.

For the QPNs (see Figure 4), the following notations and concepts are used:

- $N = (P, T, F)$ is a net with two kinds of disjoint node sets,
- a finite place set $P = \{p_0, p_1, ..., p_{m-1}\}, (m \geq 0)$,
- a finite transition set $T = \{t_0, t_1, ..., t_{n-1}\}, (n \geq 0)$,
- $F \subseteq P \times T \cup T \times P$ is the flow relation,
- $^\circ t = \{p \in P | (p, t) \in F\}$ describes the transition t input place set,
- $t^\circ = \{p \in P | (t, p) \in F\}$ describes the transition t output place set.

QPN model shown in Figure 4 corresponds to QLC represented in Figure 2 if the following relations are added.

The Toffoli (CCNOT) gate that is applied to the 3 qubits has the correspondent matrix operator:

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

The matrix operator for CNOT gate applied to the two qubits is:

$$A'_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9}$$

The NOT (Pauli-X) gate matrix operator, applied to last one qubit is:

$$A'_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{10}$$

The last two matrices are transformed, using the relation (7), resulting the mappings assigned to transitions:

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

and

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{12}$$

There are two referential systems denoted by V and Ψ that store in parallel the information in the QPN places. The quantum logic circuits (i.e., the mappings $mapV$) act on the V level (i.e., qubit registers) modifying the coefficient values $V = [(\alpha_0, \beta_0), (\alpha_1, \beta_1), \ldots, (\alpha_{g-1}, \beta_{g-1})]$, while the mappings $map\Psi$ act at the Ψ (computational) level modifying the amplitudes $[c_0, c_1, \cdots, c_{r-1}]$. For each value V there is the transformation (5) denoted by $\mathcal{T}$ that converts the V vector in $|\psi\rangle = \mathcal{T}(V)$ vector. For some pure states of $|\psi\rangle$ the transformation and the revers transformation (conversion) $V = \mathcal{T}^{-1}(|\psi\rangle)$ can be performed.

Some quantum algorithms use and have defined mapping $map\Psi$, even if at the low level they are implemented by mappings on quantum logic circuits (QLC). This requires the synthesis of QLCs that perform the mappings $map\Psi$. Other quantum algorithms have defined operations only at the qubit level. The conversion from V to Ψ is always defined by the transformation of the type $\mathcal{T}$.

Related to QPN states, the following are specified:

- Any place $p_s \in P$ has assigned a pair of natural numbers (g_s, r_s), with $r_s = 2^{g_s}$ specifying the dimensions of the tokens (i.e., vectors) $|\psi_s\rangle$ and V_s, respectively,
- $Q = \{|q_0\rangle, |q_1\rangle, \ldots, |q_{r-1}\rangle\}$ is the place quantum state set,
- $|\psi_s\rangle = \sum_{l=0}^{r-1} c_l^s |q_l\rangle$ is the first part of the token that can be set in a place p_s;
- $V_s = [(\alpha_0^s, \beta_0^s), (\alpha_1^s, \beta_1^s), \ldots, (\alpha_{g-1}^s, \beta_{g-1}^s)]$ is the second part of the token set in the place p_s,
- The marking of a place p_s is $QM(p_s) = [|\psi_s\rangle, V_s]$,
- When the token is missing in a place p, the place marking is $QM(p) = \phi$ (i.e., the empty set meaning nothing or the lack of information);
- The system quantum superposed state QS is

$$QS = [QM(p_0), QM(p_1), \ldots, QM(p_{m-1})] \tag{13}$$

and it is given by

$$QS = [|\psi_0\rangle, |\psi_1\rangle, \ldots, |\psi_{m-1}\rangle], \tag{14}$$

or

$$QS = [V_0, V_1, \ldots, V_{m-1}]. \tag{15}$$

2.5. Transition Admissibility and Execution

The QPN places can store two kinds of tokens (V token and Ψ token) and its transitions can execute two kinds of mappings ($mapV$ and $map\Psi$) depending on the available tokens in their input places.

Any output arc $(t_k, p_v) \in F$ of a transition t_k has assigned a mapping denoted $map\Psi_k^v(\ldots, |\psi_j\rangle, \ldots)$, and a mapping $mapV_k^v(\ldots, V_j, \ldots)$ with $|\psi_j\rangle$ and V_j tokens in $QM(p_j)$, $p_j \in^\circ t_k$ that transforms the information stored in its input places (i.e., Hilbert and V vectors) and sets the new token (i.e., $|\psi\rangle$ and V vectors) in its output place $p_v \in t_k{}^\circ$ according to:

$$|\psi_v\rangle = map\Psi_k^v(|\psi_i\rangle, |\psi_j\rangle); p_i, p_j \in^\circ t_k \tag{16}$$

or its counterpart

$$V_v = mapV_k^v(V_i, V_j). \tag{17}$$

When one of them is missing, the Equation (5) is used for transformation.

A *transition is enabled* for the execution of its assigned $mapV$ and/or $map\Psi$ mappings if and only if it has in its input places the corresponding V/Ψ tokens, respectively.

The *execution of an enabled transition* mapping involves the atomic logical extraction of the tokens from its input places and the injection of the tokens in its output places. When a $mapV$ mapping is executed, it is followed by the conversion and set of the $|\psi\rangle = \mathcal{T}(V)$ tokens in the output places if the $map\Psi$ mapping is not available or allowed.

An enabled *transition* t_k is executed at a moment in the time interval $(0, d_k)$ with 0 and d_k real numbers; where d_k is a very short estimated (specified) moment in time.

The start and the end of a quantum program involve:

- *init*, a method that initializes the places with quantum vectors with amplitudes from the matrix CS^0
$$QS^0 = init(CS^0), \tag{18}$$
with CS^0 an environment matrix of QS size and complex elements fulfilling the Hilbert space condition;
- *end*, a method that stops the quantum process by collapsing the current QS, extracts the information from places (i.e., the final state QS^f) and sets it to an environment matrix CS^f
$$CS^f = end(QS^f), \tag{19}$$
where CS^f is a matrix of $g \cdot m$ size with the elements in $\{0, 1\}$.

2.6. Definition of QPNs

The *definition* of QPN is:

$$QPN = (N, D, G, Map, QM^0, init, end, QM^f), \tag{20}$$

where:

- N = (P, T, F) is a net;
- $D = [d_0, d_1, \ldots, d_{n-1}]$ is a vector containing the maximum estimated delays assigned to transitions;
- $G = [g_0, g_1, \ldots, g_{m-1}]$ is a vector containing the sizes of the vectors ($V_i, i = 0, 1, \ldots,$ $m - 1$) set in the corresponding places;
- Map is the set of mappings (i.e., pair $(map\Psi_k^v(), mapV_k^v()))$ assigned to arcs linking the transitions with places; when one of them is missing, it is denoted by $null$,
- QM^0 is a matrix with initialized values (i.e., the initial quantum superposed states);
- *init* is the initialization method;
- *end* is the stop method;
- QM^f is a matrix storing the values of the final quantum superposed state (i.e., the result matrix).

There are some differences concerning the QPNs related to classical PNs:

- The same token set in a place can be used for enabling and execution of more than one transition.
- There are non-reversible transitions (that correspond to non-reversible quantum logic circuit) and reversible transitions that correspond to reversible quantum logic gates.
- Once a non-reversible transition is executed (it extracts the tokens from its input place set), no further transition can be enabled with the extracted tokens.
- The transition assessment is performed in iterations that have no duration.
- When some transitions are enabled, they are executed even if the tokens from their input place sets have already disappeared.
- If more than one transition set tokens concurrently (simultaneously) in a place, this situation leads to conflict and has to be avoided.

2.7. Analysis and Verification of QPN Models

The need of the analysis and the verification of the quantum programs behaviors is obviously in the development process, as it is known from many procedures. Petri nets can sustain these by some popular methods. In the current approach, Petri nets-based language and the reachability graphs are used.

2.7.1. Petri Net Based Language

Using the notations: '*' for sequence, '+' for alternative, '&' for concurrence and '#' for closing a loop, the behavior of a PN can be described. For example:

- $t_1 * t_2$ describes the sequential execution of the transitions t_1 and t_2;
- $t_1 + t_2$ describes the alternative for execution of the transition t_1 or t_2;
- $t_1 \& t_2 = t_1 * t_2 + t_2 * t_1$ describes the concurrent execution;
- $(t_1 * t_2)\#t_3$ describes a loop $\sigma = (t_1 * t_2 * t_3) * \sigma$.

Details of their use are provided in [3].

2.7.2. Reachability Graphs

In [3] were used the reachabilty graphs that include the states and the transitions that lead to them. Some transitions are concurrently executed, changing the system states simultaneously. Some examples of their use are provided further.

2.8. QPN Model Examples

2.8.1. Reversible Transitions and Entangled Places

Figure 5 displays a QPN model that could execute a sequence $\sigma = t_1 * t_2$ of two reversible transitions. A reversible transition should be accepted to become initially enabled even if the token in one of its input places is missing (denoted here by ϕ).

The transition t_1 concurrently executes the relations:

$$|\psi_2\rangle = A_1|\psi_1\rangle,$$
$$|\psi_1\rangle = \begin{cases} |\psi_1\rangle, & \text{if } QM(p_2) = \phi, \\ A_1^{-1}|\psi_2\rangle, & \text{otherwise} \end{cases},$$

where the unitary matrices A_1 and A_1^{-1} fulfill the relation $A_1 \cdot A_1^{-1} = I$.

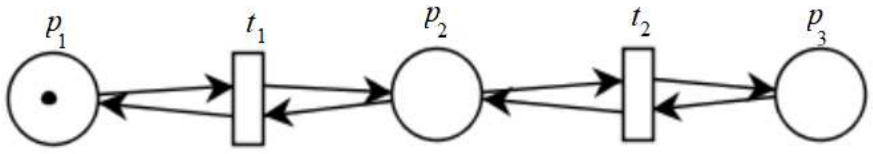

Figure 5. QPN model of a sequence of reversible transitions.

After t_1, the transition t_2 concurrently executes the relations:

$$|\psi_3\rangle = A_2|\psi_2\rangle,$$
$$|\psi_2\rangle = \begin{cases} |\psi_2\rangle, & \text{if } QM(p_3) = \phi, \\ A_2^{-1}|\psi_3\rangle, & \text{otherwise} \end{cases},$$

where the unitary matrices A_2 and A_2^{-1} fulfill the relation $A_2 \cdot A_2^{-1} = I$.

The places p_1 and p_3 are *entangled*, if the transitions t_1 and t_2 are assigned with $d_1 = 0$ and $d_2 = 0$ (i.e., zero delays). These lead to instantaneous maintenance of the relations $|\psi_3\rangle = A_2 \cdot A_1|\psi_1\rangle$ and $|\psi_1\rangle = A_1^{-1} \cdot A_2^{-1}|\psi_3\rangle$.

2.8.2. Qubit Rotations and While Loops

The QPN shown in Figure 6 describes the rotation of the qubit $|q_0\rangle$ stored in the place p_0 by the mapping assigned to t_0. The rotation with an angle θ can be determined using the circle representation displayed on Figure 7. Supposing that the rotation angle is given by a qubit $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, the angle θ is given by $\theta = \arcsin(\beta) = \arccos(\alpha)$. Performing this calculus offline, the mapping assigned to t_0 that achieves the rotation is:

$$A_1(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}. \tag{21}$$

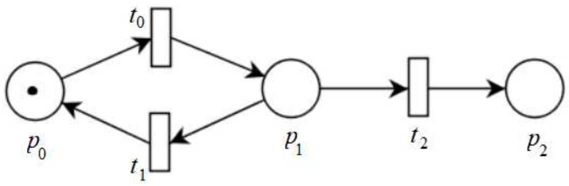

Figure 6. QPN model of a while loop.

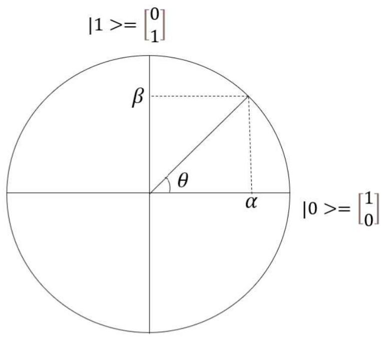
$|1>= \begin{bmatrix}0\\1\end{bmatrix}$
β
θ
α
$|0>= \begin{bmatrix}1\\0\end{bmatrix}$

Figure 7. Polar plot for representation of qubit.

The repetition of the loop until a specified state is achieved, is a more difficult task. In the current proposal the method introduced by [66] is considered to be used. As a consequence, it is supposed that a loop can be admittedly executed with a specified robustness. The measurement performed at each iteration, for predicate condition of exiting the loop, diminishes the qubits state with a value smaller than a specified and accepted one.

The QPN model displayed in Figure 6 executes the sequence:

$$\sigma = t_0 * (t_1 \& t_2) * \sigma. \tag{22}$$

The while loop is determined by t_1 that executes:

$$t_1 : \text{IF } QM(p_1) > \epsilon \text{ THEN } |\psi_0\rangle = A_1|\psi_1\rangle \text{ END}. \tag{23}$$

This involves the weak measurement of $|\psi_1\rangle$ (see details in [66]) that ends the loop execution when the condition is not met.

The QPN example displayed in Figure 8 corresponds to a mapping that rotates the qubit $|q_o\rangle = \alpha_o|0\rangle + \beta_o|1\rangle$ stored in the place p_0 with an angle given by the control qubit $|q_c\rangle = \alpha_c|0\rangle + \beta_c|1\rangle$ stored in the place p_1, resulting the qubit $|q_r\rangle = \alpha_r|0\rangle + \beta_r|1\rangle$ injected in the place p_2. This requires the synthesis of a QLC that performs the unitary transformation $(\alpha_r, \beta_r) = map((\alpha_0, \beta_0), (\alpha_c, \beta_c))$.

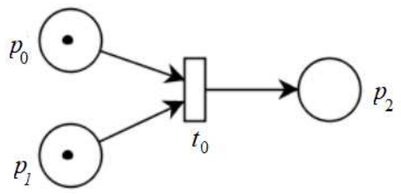

Figure 8. QPN model of a qubit rotation depending on another one.

Let be the above $map(.,.)$ given by $A_1(\theta)$ defined in (21) with $\theta = \arctan(\frac{\beta_c}{\alpha_c})$. This sets in p_2 the result of the rotation of $|q_0\rangle$ according to the amplitudes of $|q_c\rangle$.

Another possibility to implement the model displayed in Figure 8 is to assign the vectors $|\psi_0\rangle$, $|\psi_1\rangle$ and $|\psi_2\rangle$ to the QPN places. If the mapping assigned to transition t_0 is $|\psi_2\rangle = |\psi_0\rangle \otimes |\psi_1\rangle$, this manner of multiplication leads to increasing the dimension of the vector set in p_2 to $2^{g_2} = 2^{g_0} \cdot 2^{g_1}$.

If $g_1 = 1$, then the mapping can be correctly defined by $|\psi_2\rangle = M^+ \cdot (A_1(\theta) \otimes I \cdot |\psi_0\rangle)$, where I is the corresponding identity matrix. M^+ can be defined to perform an increment circular shift:

$$M^+ = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ . & . & . & . & \cdots & . & . & . \\ . & . & . & . & \cdots & . & . & . \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \end{bmatrix}. \tag{24}$$

Similarly, a matrix M^- can be defined, which performs a decrement circular shift.

The problem can be extended for the case when $g_1 = 2$ and the previous places model vectors of qubits. If $A_1(\theta_0)$ and $A_1(\theta_1))$ perform the rotations of two independent qubits, the matrix:

$$A_2(\theta_0, \theta_1) = \begin{bmatrix} A_1(\theta_0) & \mathbf{0} \\ \mathbf{0} & A_1(\theta_1) \end{bmatrix}, \tag{25}$$

achieves the rotation of a quantum vector (i.e., register) of two qubits.

2.8.3. Quantum Discrete Random Walker on a Circle

The chosen examples concern the Quantum Discrete Random Walker (QDRW) that moves on a circle or on a sphere (see Figure 9). They can be oriented or un-oriented. The specified 8 discrete positions on the circle can be coded with three qubits composing a vector $V = [(\alpha_0, \beta_0), (\alpha_1, \beta_1), (\alpha_2, \beta_2)]$ or by referring to the quantum states for each position $|\psi_X\rangle = \sum_{k=0}^{7} c_k^x |q_k\rangle$.

The walker random moves are decided by throwing a coin that can be modeled by Hadamard matrix (Equation (6)).

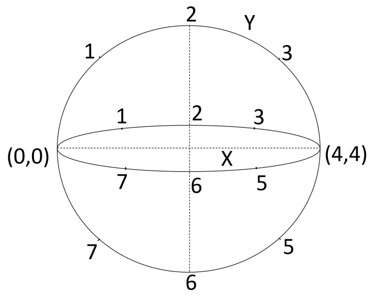

Figure 9. QDRW movement space.

The QDRW model on a circle can be described in the Hilbert space $\mathcal{H} = \mathcal{H}_X^8$ by the equation:

$$|\psi_x(\tau+1)\rangle = SH(H \bigotimes I \cdot |\psi_x(\tau)\rangle) \tag{26}$$

where $SH(...)$ denotes the shift operator applied to its argument and τ is the clock tick. SH moves the walker to the right if the decision operator (i.e., the coin) provides "+" and to the left if the result is negative. SH is implemented by a permutation matrix (that has only one '1' on each column or line) that performs the walker evolution. SH^+ achieves a positive move (i.e., increase the position), while SH^- determines a negative move (decrease the position value).

Denoting by $\varrho = \frac{1}{\sqrt{2}}$, the application of H to a qubit provides:

$$H \cdot |q_0\rangle = \varrho|0\rangle + \varrho|1\rangle$$
$$H \cdot |q_1\rangle = \varrho|0\rangle - \varrho|1\rangle$$

Denoting with $X = [c_0^x, c_1^x, \ldots, c_7^x]^T$, the probability of the QDRW to be in the position $x = k, (0 \leq k \leq 7)$ at a moment τ is $\pi_k(\tau) = |c_k(\tau)|^2$.

The probability that QDRW follows a trajectory described by the sequence $\sigma = i * j * k$ is $\Pi_\sigma = \pi_i(\tau) * \pi_j(\tau+1) * \pi_k(\tau+2)$.

The probability that QDRW hits a target $x = k$ during a discrete time interval $\theta = [a, b]$ is given by $\Pi_{x=k;\theta} = \sum_{\tau=a}^{b} |c_k(\tau)|^2$.

The oriented Quantum Discrete Random Walker (QDRW) QPN model is represented in Figure 10. Its state $W = (O, X)$ is described in the Hilbert space $\mathcal{H} = \mathcal{H}_O \otimes \mathcal{H}_X^8$ by the equation:

$$|\psi_w(\tau)\rangle = (\alpha(\tau)|0\rangle + \beta(\tau)|1\rangle) \bigotimes \sum_{k=0}^{7} c_k(\tau)|q_k\rangle \tag{27}$$

with $\alpha(\tau)$ and $\beta(\tau)$ the complex coefficients needed for orientation fulfilling the Hilbert condition, and $c_k, i = 0, \ldots, r$ complex numbers from the set $\mathbb{C}$ fulfilling the relation: $\sum_{k=0}^{r-1}(|\alpha(\tau)c_k(\tau)|^2 + |\beta(\tau)c_k(\tau)|^2) = 1$ in any moment τ.

The place p_2 models the QDRW oriented toward increasing the position (i.e., state X_0, while the place p_5 corresponds to its orientation toward decreasing the position (i.e., X_1). Throwing the coin (i.e., applying the operator H) determines the change of orientations that affect the moves performed by shift operators. The QDRW dynamics is achieved by the mappings assigned to transitions.

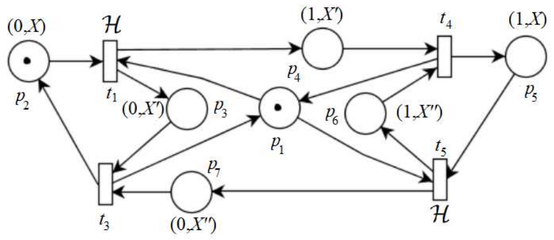

Figure 10. QPN of oriented QDRW on a circle.

The model executes the sequence: $\sigma = ((t1\&t2) * (t3\&t4))^k$. The implementation requires three qubits for the walker's position and one for the orientation.

The mappings assigned to transitions are:

- $t_{1\_3} : X'^0(\tau+1) = \varrho \cdot X^0(\tau)$
- $t_{1\_4} : X'^1(\tau+1) = \varrho \cdot X^0(\tau)$
- $t_{2\_6} : X''^1(\tau+1) = -\varrho \cdot X^1(\tau)$
- $t_{2\_7} : X''^0(\tau+1) = \varrho \cdot X^1(\tau)$
- $t_3 : X^0(\tau+1) = SH^+(X'^0(\tau+1) + X''^0(\tau+1))$
- $t_4 : X^1(\tau+1) = SH^-(X'^1(\tau+1) + X''^1(\tau+1))$

The probability that QDRW be in a specified position x_k at a time moment is given by the sum of the probabilities given by values stored in the places p_2 and p_5:

$$\pi_k(\tau) = |c_k^0|^2 + |c_k^1|^2, \tag{28}$$

where c_k^0 and c_k^1 are the elements of the vectors X^0 and X^1, respectively. These probabilities are displayed in Figure 11.

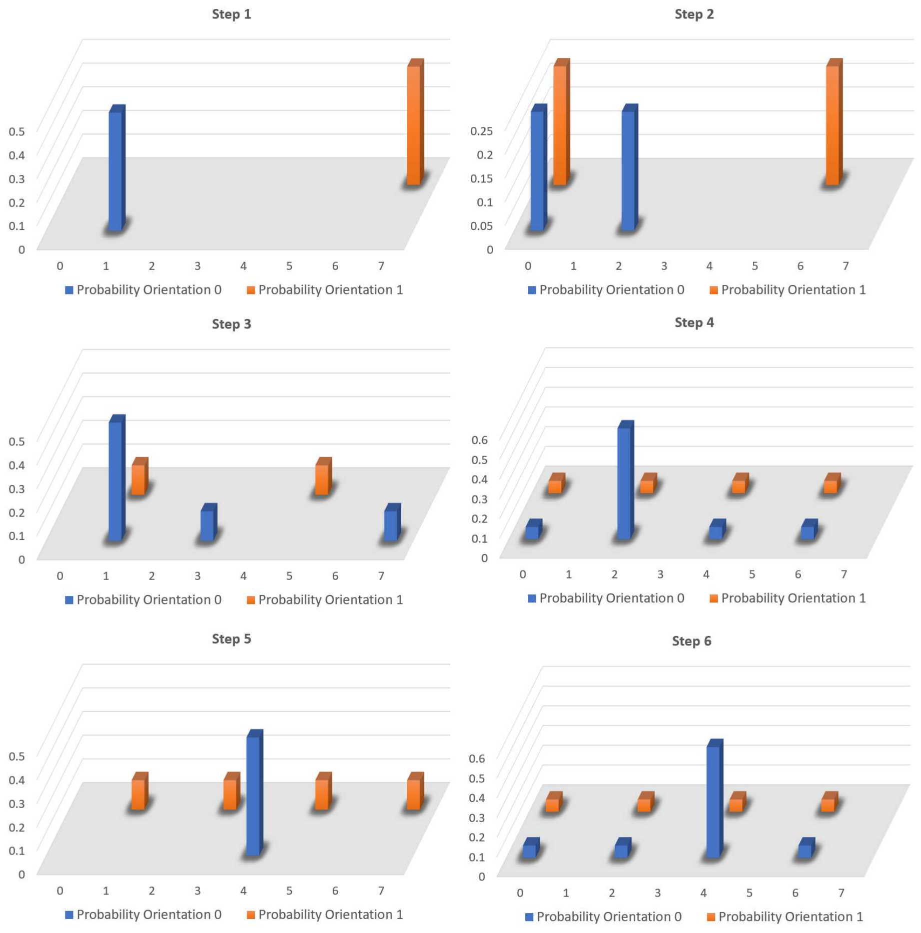

Figure 11. Probabilities of oriented QDRW on a circle.

For the QPN analysis of oriented QDRW on a circle its reachability graph is constructed as can be seen in Table 1. Instead of the quantum state, QRS is used here.

Table 1. Reachability graph of QPN from Figure 10.

p_1	p_2	p_3	p_4	p_5	p_6	p_7	**Transition**
$*$	X^0	ϕ	ϕ	ϕ	ϕ	ϕ	t_1
ϕ	ϕ	X'^0	X'^1	ϕ	ϕ	ϕ	$t_3\&t_4$
$*$	X^0	ϕ	ϕ	X^1	ϕ	ϕ	$t_1\&t_2$
ϕ	ϕ	X'^0	X'^1	ϕ	X''^1	X''^0	$t_3\&t_4$
...	...	...	...	...	...	...	...

2.8.4. Quantum Random Walker on a Sphere

Figure 9 represents the sphere (like the Earth globe) used for the walker's move and Figure 12 shows its surface displayed on two dimensions. The walker initial position is (x_0, y_0).

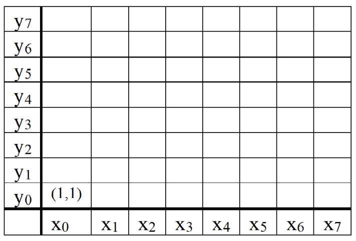

Figure 12. Sphere surface deployment on two dimensions.

The system composed of *oriented QDRW moving on a sphere* is described in the Hilbert space:

$$\mathcal{H} = \mathcal{H}_O^4 \bigotimes \mathcal{H}_X^8 \bigotimes \mathcal{H}_Y^8 \tag{29}$$

with the vectors $|\psi_X\rangle = \sum_{k=0}^{7} c_k^x \cdot |q_k\rangle$ and $|\psi_Y\rangle = \sum_{k=0}^{7} c_k^y \cdot |q_k\rangle$ describing the position on axes X and Y, respectively. The walker state is extended with its orientation on the both axes: $|\psi_X^O\rangle = (\alpha_0|0\rangle + \beta_0|1\rangle) \otimes \sum_{k=0}^{7} c_k^x \cdot |q_k\rangle$ and $|\psi_Y^O\rangle = (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes \sum_{k=0}^{7} c_k^y \cdot |q_k\rangle$.

The action of the coin vector $|\psi_K\rangle = \sum_{k=0}^{3} c_k^C \cdot |q_k\rangle$ that modifies the walker orientation is given by the matrix G (according to [67]):

$$G = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}. \tag{30}$$

The oriented QDRW movements are given by the formula:

$$[|\psi_X^O(\tau+1)\rangle, |\psi_Y^O(\tau+1)\rangle]^T = SH(G \cdot [|\psi_X^O(\tau)\rangle, |\psi_Y^O(\tau)\rangle]^T), \tag{31}$$

where the square brackets are used for describing the construction of vectors.

For this problem is conceived the QPN model displayed in Figure 13. The places p_0, p_1, p_2 and p_3 model walkers states $((X^0, Y^0), (X^1, Y^1), (X^2, Y^2), (X^3, Y^3))$ oriented to the right, left, up and down, respectively.

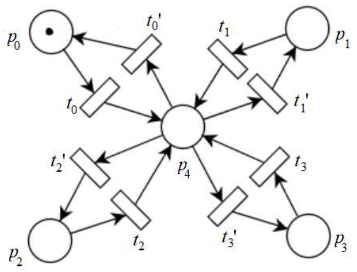

Figure 13. QPN model of oriented QDRW on a sphere.

The place meanings are displayed in Table 2. The initial state is: $X^0 = Y^0 = [(\varrho, \varrho), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0), (0,0)]$.

Table 2. Place meanings of QPN from Figure 13.

***p*0**	***p*1**	***p*2**	***p*3**	**Place Array *p*4**
(X^0, Y^0)	(X^1, Y^1)	(X^2, Y^2)	(X^3, Y^3)	$(X_{00}, Y_{00}).(X_{01}, Y_{01}), \cdots, (X_{33}, Y_{33})$

The model executes the sequence: $\sigma = t_0 * ((t_0'\&t_1'\&t_2'\&t_3') * (t_0\&t_1\&t_2\&t_3))^k$. The implementation requires three qubits for X walker's position, another three qubits Y and two qubits for the walker orientations on X and Y.

The mappings assigned to the transitions are:

- $t_0 : X_{00} = -\frac{1}{2}X^0; Y_{00} = -\frac{1}{2}Y^0; X_{01} = \frac{1}{2}X^0; Y_{01} = \frac{1}{2}Y^0; \cdots ; X_{03} = \frac{1}{2}X^0; Y_{03} = \frac{1}{2}Y^0;$
- $t_1 : X_{10} = \frac{1}{2}X^1; Y_{10} = \frac{1}{2}Y^1; X_{11} = -\frac{1}{2}X^1; Y_{11} = -\frac{1}{2}Y^1; \cdots ; X_{13} = \frac{1}{2}X^1; Y_{13} = \frac{1}{2}Y^0;$
- $t_2 : X_{20} = \frac{1}{2}X^2; Y_{20} = \frac{1}{2}Y^2; X_{21} = \frac{1}{2}X^2; Y_{21} = \frac{1}{2}Y^2; \cdots ; X_{23} = \frac{1}{2}X^2; Y_{23} = \frac{1}{2}Y^2;$
- $t_3 : X_{30} = \frac{1}{2}X^3; Y_{30} = \frac{1}{2}Y^3; X_{31} = \frac{1}{2}X^3; Y_{31} = \frac{1}{2}Y^3; \cdots ; X_{33} = -\frac{1}{2}X^3; Y_{33} = -\frac{1}{2}Y^3;$
- $t'_0 : X^0 = SH^+(X_{00} + X_{10} + X_{20} + X_{30}); Y^0 = SH^+(Y_{00} + Y_{10} + Y_{20} + Y_{30});$
- $t'_1 : X^1 = SH^-(X_{01} + X_{11} + X_{21} + X_{31}); Y^1 = SH^+(Y_{01} + Y_{11} + Y_{21} + Y_{31});$
- $t'_2 : X^2 = SH^+(X_{02} + X_{12} + X_{22} + X_{32}); Y^2 = SH^-(Y_{02} + Y_{12} + Y_{22} + Y_{32});$
- $t'_3 : X^3 = SH^-(X_{03} + X_{13} + X_{23} + X_{33}); Y^3 = SH^-(Y_{03} + Y_{13} + Y_{23} + Y_{33});$

For the QPN analysis of oriented QDRW on a sphere, its reachability graph is displayed in Table 3.

Table 3. Reachability graph of QPN from Figure 13.

***p*0**	***p*1**	***p*2**	***p*3**	***p*4**	**Transition**
(X^0, Y^0)	ϕ	ϕ	ϕ	ϕ	$t0$
ϕ	ϕ	ϕ	ϕ	$(X_{00}, Y_{00}) \cdots (X_{03}, Y_{03})$	$t'_0 \& t'_1 \& t'_2 \& t'_3$
(X^0, Y^0)	(X^1, Y^1)	(X^2, Y^2)	(X^3, Y^3)	ϕ	$t_0 \& t_1 \& t_2 \& t_3$
ϕ	ϕ	ϕ	ϕ	$(X_{00}, Y_{00}) \cdots (X_{33}, Y_{33})$	$t'_0 \& t'_1 \& t'_2 \& t'_3$
...	...	...	...	...	...

The probability of the oriented QDRW to be at the moment τ in the position $X = x_i$ and $Y = y_j$ is given by the coefficients of the vectors $|\psi_X(\tau)\rangle$ and $|\psi_Y(\tau)\rangle$ at the moment τ:

$$\pi_X(\tau; i) = |c_i^{X^0}(\tau)|^2 + |c_i^{X^1}(\tau)|^2 + |c_i^{X^2}(\tau)|^2 + |c_i^{X^3}(\tau)|^2, \tag{32}$$

$$\pi_Y(\tau; j) = |c_j^{Y^0}(\tau)|^2 + |c_j^{Y^1}(\tau)|^2 + |c_j^{Y^2}(\tau)|^2 + |c_j^{Y^3}(\tau)|^2, \tag{33}$$

$$\pi_{XY}(\tau; i, j) = \pi_X(\tau; i) \cdot \pi_Y(\tau; j). \tag{34}$$

The needed information required for these calculations is stored in the places p_0, p_1, p_2 and p_3. The results of simulation can be seen in Figure 14.

2.9. Quantum Discrete Controlled Walker (QDCW)

The previous oriented walker's randomness (given by the generalized coin) is replaced by a matrix denoted by Γ that influences its move directions. The QDCW move equation becomes:

$$[|\psi_X^O(\tau+1)\rangle, |\psi_Y^O(\tau+1)\rangle] = SH(G \cdot \Gamma(\tau)[|\psi_X^O(\tau)\rangle, |\psi_Y^O(\tau)\rangle]) \tag{35}$$

with

$$\Gamma(\tau) = A(\theta_1, \theta_2) \tag{36}$$

and G the previous diffusion matrix. The angles θ_1 and θ_2 determine the move directions. Again, the shift operator SH determines the QDCW moves toward these angles.

The control problem consists of the application of Γ that leads the QDCW from a given initial state (i.e., position and orientation) to some specified target points or region.

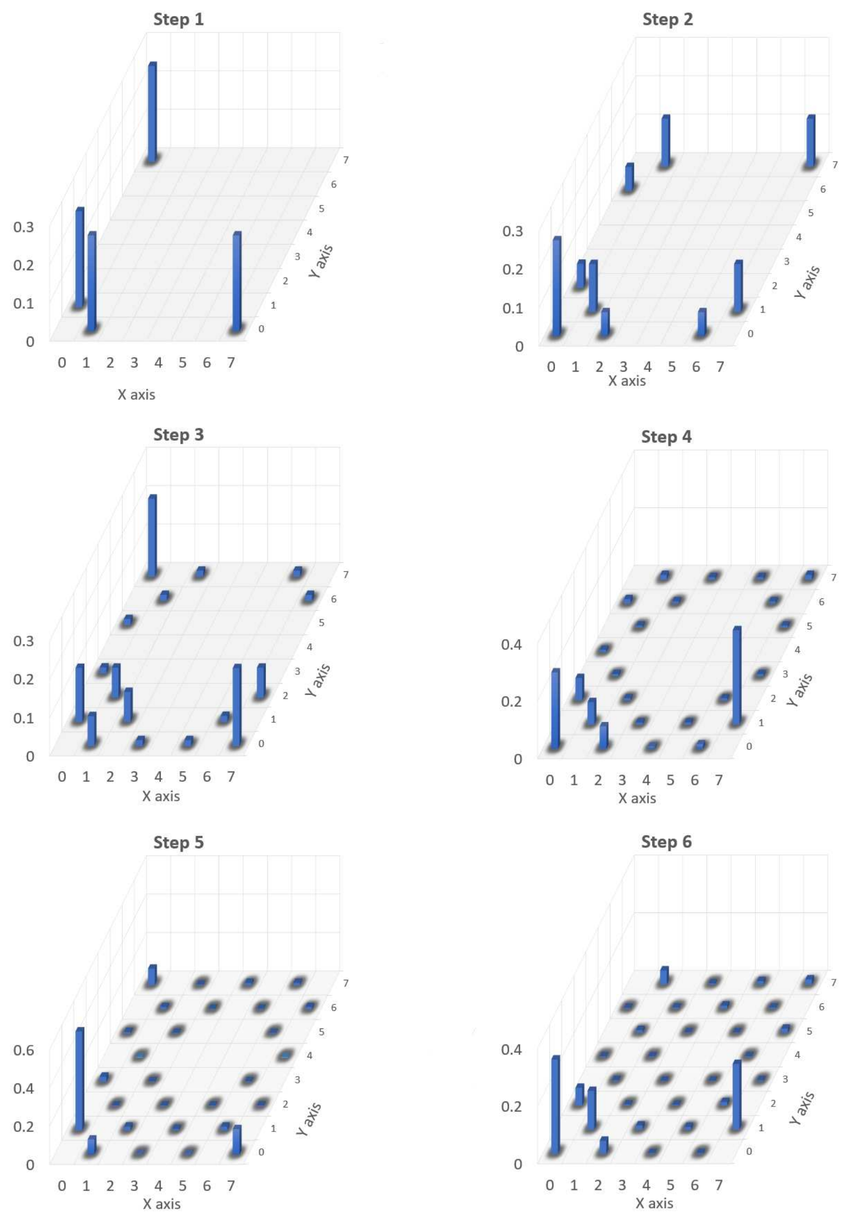

Figure 14. Probability representation of oriented QDRW moves on a sphere.

2.10. Classic and Quantum Computers Connection

The quantum algorithms are not executed independently, but in collaboration with classic implemented algorithms.

Figure 15 shows a composition of an Object Enhanced Time Petri Net (OETPN, see [68]) model that interacts with a QPN model. OETPN receives demands from operator through the classic place cp_2 and interacts with QPN through the interface (classic-quantum place) cqp. This provides the information used for the first steps of the quantum program. OETPN receives the quantum computation results through the interface (quantum-classic place) qcp.

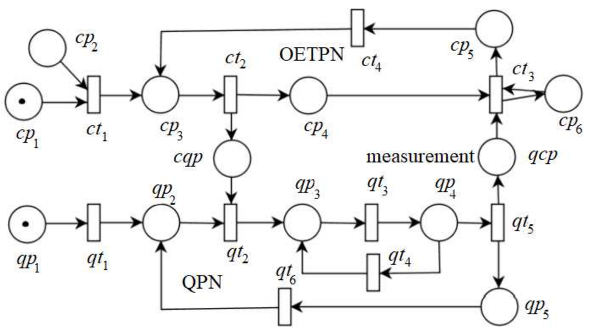

Figure 15. Classic-quantum application structure.

According to [32] a qubit $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ observation or measurement can be modeled by the wave function collapse as:

$$\begin{cases} x = 0, & \text{if } |\alpha|^2 \leq pr(x'), \\ x = 1, & \text{if } |\alpha|^2 > pr(x'), \end{cases} \tag{37}$$

where $pr(x')$ is a random number in the range $[0,1)$ used for the assessing of x.

QPN is initialized by the transition qt_1 in the pure state $|00\ldots0\rangle$ that is switched by qt_2 in the desired initial state V or $|\psi\rangle$ state in qp_3 using the information provided by cqp. The mapping(s) of qt_3 applies the QLC operations. If the exit condition is not fulfilled, the quantum calculations are repeated after the reaching of the qp_3 inserted by the transition qt_4. When the exit condition is reached, the transition qt_5 sets the measured value of qp_4 by collapsing on specified axis.

Approximately $1/\epsilon^2$ measurements are required to get the results with the precision ϵ. This means to repeat the quantum process with the same initialization values. OETPN can request to repeat the QPN execution by reloading the initial condition, until the expected statistical results are obtained.

The *QPN* places do not necessarily independently represent the quantum processor qubits that are described by the entire QPN marking. The QPN places describe the quantum program state. All the transitions' mappings are implemented at the lower level by QLCs.

2.11. Quantum Evolutionary Systems (QES)

The extension of classical ES to Quantum Evolutionary Systems (QES) concerns the replacing of the genes of classical genomes with qubits. This makes the description of an infinite number of individuals in the same instantiation of the model possible. The classical genetic operators have to be adapted to the new form of information encryption.

The individual evaluation, which in the classical computation requires the separate simulation of each of them, can be performed simultaneously. The reading of evaluations performed with different parameters can extract the information for individual selection and further on for their improvement.

Quantum control is an effective tool to drive the quantum system to a given target state and thus manipulate the dynamics of a quantum system [61]. The main goal is to control a multilevel system from an initial state to a given target state. This can solve problems related to quantum control strategies based on supervised machine learning to suppress the quantum noise in an open quantum system.

Quantum approach can be used in GA by:

- Quantum Inspired Genetic Algorithms (QIGA) where the simulation and improvement are implemented on classical computers;
- Hybrid Classic Quantum GA (HCQGA) where the simulations are performed by QC, but the improvements are determined by tasks implemented on classical computers;
- CQGA (Complete Quantum Genetic Algorithm) where the individual simulations and improvements are performed by tasks implemented completely on QC.

2.12. Quantum Genetic Algorithms

Genetic Algorithms contain:

- The genome specification that is used for construction of the genotypes,
- The individual evaluation that consists of modeling the individual behaviors and assessing their performances,
- The individual selection for reproduction,
- The individual improvements that are performed randomly based on the previous generation performance and the use of some genetic operators,
- The stopping criteria that uses either a number of tested generations reaching a performances over a desired threshold, or just the execution is exceeding a specified duration or a specified number of generations without relevant improvements.

Figure 16 shows the evolutionary process performed by a GA. It executes the sequence $\sigma = t_0 * (((t_1 * t_2)\&t_4)\#(t_3 * t_5))^K * t_6$ where K represents the number of iterations performed until the result is obtained.

Generally, the ESs need to properly solve the evaluations of individuals and their improvement. Both of them should be adapted to the problems, otherwise the targets are not reached. Due to the problem complexities, there are no clear and proven rules for the general construction of fitness functions and individual evolution operators. There are two main approaches for individual evolution (i.e., improvements) of QGAs: one uses only the mutation and the other uses the crossover before performing the mutation. The first approach uses the best individual from the previous generation as a target for random genotype improvement, while the second selects the individuals for evolution according to their performances and randomly interchanges the two individual genotypes.

For achieving the solution, in [69], the authors propose a QGA implementation that does not need the crossover involving two individuals. The justification is that all the individuals are simulated simultaneously. More recently, [57] includes and justifies the use of crossover.

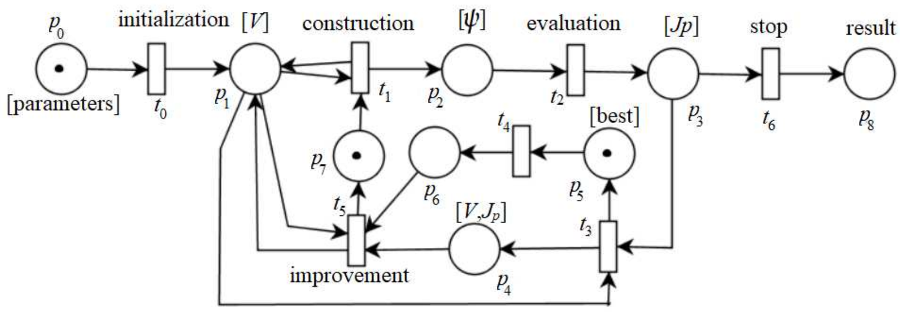

Figure 16. OETPN model of an evolutionary process.

The current approach concerns the control of the Quantum Discrete Walker with the model shown in Figure 13. The individuals of GA determine the mappings assigned to transitions $t_0, t_1, t_2, t_3, t'_0, t'_1, t'_2$, and t'_3. The Classic GA (CGA) problem usually involves the finding in each intersection (node and/or time) of the walker directions such that it reaches the targets avoiding the traps. Due to the fact that the walker can start from different initial points, the search should be repeated from all of them.

For the Quantum Discrete Controlled Walker (QDCW), the problem can be defined in two manners: one solves only the change of direction and the other performs the move length (i.e., $0, 1, 2, \dots$) in the chosen direction.

The efficiency of QGA is significantly better than the conventional genetic algorithm [49]. The classic Genetic Algorithm (CGA) has to execute (simulate) many individuals for their assessment one by one. QC (quantum computing) has the advantage that it can simulate simultaneously a large number of individuals. QC uses quantum variables that can cover by superposition the entire search domain. QGAs (Quantum Genetic Algorithms) have to represent the individuals by quantum vectors. The individual modifications fulfill the Hilbert condition by using mappings involving unitary matrices.

The transition t_0 of the model shown in Figure 16 creates in the place p_1 the genotypes pool using the initial information stored in the place p_0. The genotype V is used for the construction of the individual ψ. For the walker control problem, the solution is obtained by replacing the previous G matrix (Equation (30)) with $G \cdot \Gamma$, where Γ is a unitary matrix ($\Gamma = [\gamma_{j,k}]_{j,k=0,1,2,3}$) that has the role to diminish the randomness and to lead the walker toward the targets. The genes are $|\gamma_0\rangle = \alpha_0|q_0\rangle + \beta_0|q_1\rangle$ and $|\gamma_1\rangle = \alpha_1|q_0\rangle + \beta_1|q_1\rangle$ that, for implementation reasons, are changed in the vector $V_j = [(\alpha_0^j, \beta_0^j), (\alpha_1^j, \beta_1^j)]$ describing the quantum individual genotype.

The execution of the mapping assigned to transition t_2 evaluates the individuals using the model displayed in Figure 13. The walkers described by $[\psi - o, \psi_x, \psi - y]$ are equally initialized $V_0^w = [(\varrho, \varrho) \cdots (\varrho, \varrho)]$ to occupy the entire walker's space:

$$\mathcal{H} = \mathcal{H}_O^4 \bigotimes \mathcal{H}_X^8 \bigotimes \mathcal{H}_Y^8 \tag{38}$$

The individual performances are set in the place p_3. The transition t_3 selects the individuals for evolution setting them in p_4, linking their genotypes with the performances. For improvement, the best individuals from the previous generation are stored in p_5. The transition t_5 performs the individual improvement (i.e., population evolution). The evolution continues until a stop condition is fulfilled, and then the transition t_6 is executed, providing the final results in the place p_8.

In conclusion, all the possible initial states of the controlled system are evaluated simultaneously based on the walker's superposition. The transitions t_1, t_2, t_3 and t_5 can perform their activities in parallel for the current population.

Problems of optimal control of the QDCW can be defined in two manners. One consists of the application of control signals at each tick for direction modification, and the other one involves the syntheses of an environment that leads to a kind of deflectors set in each intersection. The deflectors direct the QDCW towards the desired targets avoiding the undesired traps. The current approach concerns the first manner.

The QPN model of the QDCW that solves the first approach is shown in Figure 17. As can be seen, the parallel simulations of QDCW, the controller (replacing in each moment of time the former Γ) and the individual evaluation (i.e., fitness) is governed by the Clepsydra ticks that will stop the execution when the experiment duration expires.

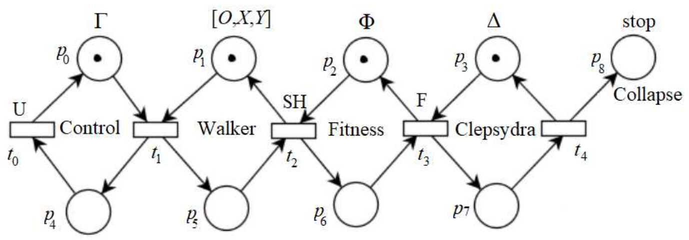

Figure 17. Quantum individual evaluation.

QDCW can discretely move on a sphere with the above presented structure. It has to reach some target points $T_g = \{(x_g^a, y_g^a), (x_g^b, y_g^b), \dots\}$ avoiding some trap points $T_p = \{(x_p^a, y_p^a), (x_p^b, y_p^b), \dots\}$.

Figure 18 shows the QDCW problem with initial walker state $(1,1)$, $T_g = \{(5,5)\}$ and $T_p = \{(2,4), (3,3), (4,2)\}$. The sequence $0*1*2*3*4*5*6*7$ represents an obtained (by CGA) optimal trajectory.

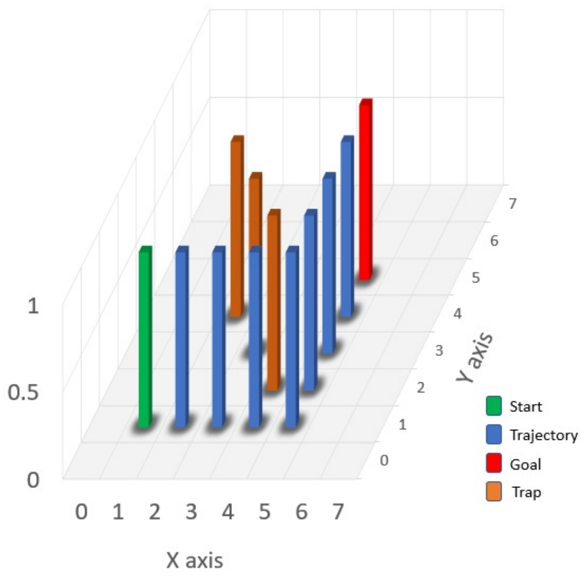

Figure 18. Representation of the targets and traps of the QDCW problem solved by a classic GA.

The walker state is described by superposition of the composed vectors:

$$[|\psi_o(\tau)\rangle, |\psi_x(\tau)\rangle, |\psi_y(\tau)\rangle]^T. \tag{39}$$

QDCW movement for 8 clock ticks is represented by the sequence:

$$\mathcal{M} = [|\psi_o(0)\rangle, |\psi_x(0)\rangle, |\psi_y(0)\rangle]^T * [|\psi_o(1)\rangle, |\psi_x(1)\rangle, |\psi_y(1)\rangle]^T * \cdots \\ *[|\psi_o(8)\rangle, |\psi_x(8)\rangle, |\psi_y(8)\rangle]^T. \tag{40}$$

2.12.1. Fitness Functions

For the classic GA, an efficient *fitness function* is constructed based on assessing the trajectory cost on each point. When QDCW reaches a point (i, j) with the probability π_{ij}, the cost to be in that position is:

$$Cost(i, j, \tau) = \begin{cases} 20 \cdot \pi_{ij}, & \text{if } (i, j) \in T_g, \\ -20 \cdot \pi_{ij}, & \text{if } (i, j) \in T_p, \\ \pi_{ij}, & \text{otherwise.} \end{cases} \tag{41}$$

The cost of the QDCW at a moment of time τ is:

$$Cost(|\psi_x(\tau)\rangle, |\psi_y(\tau), \tau\rangle\rangle) = \sum_{0 \le i \le 7} \sum_{0 \le j \le 7} Cost(i, j, \tau) \tag{42}$$

The move performance is assessed with the formula:

$$J_p = \sum_{\forall[|\psi_o(\tau)\rangle, |\psi_x(\tau)\rangle, |\psi_y(\tau)\rangle] \in \sigma}^{\exists \tau \leqslant 15; [|q_k^X\rangle, |q_k^Y\rangle] \in T_g; |c_k^X(\tau)|^2 \ge \eta; |c_k^Y(\tau)|^2 \ge \eta;} Cost(|\psi_x(\tau)\rangle, |\psi_y(\tau), \tau\rangle\rangle) \tag{43}$$

with η a specified real value considered acceptable for the walker to be in a target point.

As a consequence, if the sequence σ includes points $[|\psi_x(\tau_i)\rangle, |\psi_y(\tau_i)\rangle] \in T_g$ and $[|\psi_x(\tau_i)\rangle, |\psi_y(\tau_j)\rangle] \in T_p$ such that $\tau_i < \tau_j$ (i.e., τ_i is precedent to τ_j), then

$$Cost([|\psi_x(\tau_i)\rangle, |\psi_y(\tau_i)\rangle]) = 0. \tag{44}$$

Moreover, when the walker hits a target point with a probability greater than η, its move is not assessed further.

For a quantum approach of QGA, the current research used the following *quantum fitness function*. Let $\psi_{xy} = |\psi_x\rangle \otimes |\psi_y\rangle$ be the composed vector providing the information related to the walker's position (i.e., the amplitude to be there) and $\Psi_{xy} = |\psi_{xy}\rangle\langle\psi_{xy}|$ its density matrix. The diagonal elements of this matrix $\Psi_{xy}[k,k] = c_i^x(\tau)^2 c_j^y(\tau)^2; k = 0,\dots,15;$ $i = 0,\dots,7; i = 0,\dots,7;$ provide the probability of the walker to be in the position (i,j).

The set T_n is used to denote the positions when the walker is not on T_g or T_p, while T_d (meaning degenerated) denotes the case when the positions are simultaneously target and trap.

Let the values $\Psi^n(\tau), \Psi^g(\tau), \Psi^p(\tau)$ and $\Psi^d(\tau)$ be calculated by the formulas:

$$\Psi^n(\tau) = \sum_{(i,j)\in T_n} \Psi_{xy}[i,j] \tag{45}$$

$$\Psi^g(\tau) = \sum_{(i,j)\in T_g} \Psi_{xy}[i,j] \tag{46}$$

$$\Psi^p(\tau) = \sum_{(i,j)\in T_p} \Psi_{xy}[i,j] \tag{47}$$

$$\Psi^d(\tau) = \sum_{(i,j)\in T_d} \Psi_{xy}[i,j] \tag{48}$$

They will be used to assess the probabilities of QDCW to be in the mentioned partition. Obviously $\Psi^n(\tau) + \Psi^g(\tau) + \Psi^p(\tau) + \Psi^d(\tau) = 1$ evaluates the probability of QDCW to be in the current moment of time in the mentioned partition, but the individual evaluation needs the assessment of its entire trajectory. The evaluation is given by a dynamic quantum fitness function Φ composed of two parts (Φ^0, Φ^1) moving toward positive and negative (respectively) depending on the superposition of QDCW. Φ has orientation (stored by one qubit) and position (stored by 16 qubits).

The orientation is given by:

$$|O^\Phi(\tau+1)\rangle = A_1(\theta) \cdot H \cdot |O^\Phi(\tau)\rangle \tag{49}$$

where $A_1(\theta)$ is given by the formula (21) with the angle $\theta = \arcsin(\Psi^g - \Psi^p)$, and H is the Hadamard matrix.

Denoting by $a_0 = \rho(\cos(\theta) - \sin(\theta))$ and $a_1 = \rho(\cos(\theta) + \sin(\theta))$, the quantum fitness function is calculated by:

$$\Phi^0(\tau+1) = SH^+(a_0 \cdot \Phi^0(\tau) + a_1 \cdot \Phi^1(\tau)) \tag{50}$$

$$\Phi^1(\tau+1) = SH^-(a_1 \cdot \Phi^0(\tau) - a_0 \cdot \Phi^1(\tau)) \tag{51}$$

It can be seen that:

- If $\Psi^g = \Psi^p$, Φ moves equally toward positive and negative,
- If $\Psi^g > \Psi^p$, Φ moves more toward positive than negative,
- If $\Psi^g < \Psi^p$, Φ moves more toward negative than positive.

The performance is given by:

$$\begin{aligned} |\Phi^0(15)\rangle &= \sum_{k=0}^{15} c_k^{\Phi^0} \cdot |q_k\rangle \\ J_p &= [c_0^{\Phi^0}, c_1^{\Phi^0}, \dots, c_{15}^{\Phi^0}] \end{aligned} \tag{52}$$

with $c_k^{\Phi^0}$ $(k = 0, 1, \cdots, 15)$ the amplitudes of Φ^0.

All the individuals and their assessments are simulated simultaneously during the specified time horizon (see the model represented in Figure 17). It continues with Grover's algorithm that provides the best individual and its fitness.

2.12.2. Controller Genome

Regarding Figure 17, the first manner has to find the value of $\Gamma(0)$ and the unitary matrix U that modify the control signals according to the formula:

$$\Gamma(\tau+1) = U \cdot \Gamma(\tau) \tag{53}$$

The matrix $U = A_2(\theta_0^u, \theta_1^u)$ is constructed based on the relation (25).

The *controller's genome* is: $[|qb_0^u\rangle, |qb_1^u\rangle, |qb_0^\gamma\rangle, |qb_1^\gamma\rangle]$ where the vectors $|qb_0^\gamma\rangle$ and $|qb_1^\gamma\rangle$ correspond to the value of $\Gamma(0)$.

Γ changes the QDCW orientation according to the relation:

$$|\psi_o(\tau+1)\rangle = G \cdot \Gamma(\tau) \cdot |\psi_o(\tau)\rangle \tag{54}$$

In conclusion, the GA has the task of finding the angles $\theta_0^\gamma, \theta_1^\gamma, \theta_0^u, \theta_1^u$ that lead to maximum performance. For the current problem, an individual genome V_i is denoted by

$$V_i = (\theta_0^\gamma, \theta_1^\gamma, \theta_0^u, \theta_1^u)_i \tag{55}$$

The population $\mathcal{P}^w$ of the generation w is:

$$\mathcal{P}^w = [V_i]_{i=0,1,\ldots,pd-1} \tag{56}$$

where pd is the population dimension. The individual implementation requires $2+2$ qubits, allowing the simultaneous simulation of the entire population. Another 8 qubits are needed for implementing the QDCW.

2.12.3. Genetic Operators

The *individual (genotype) improvement* introduced in [49] is used for the purpose of avoiding the premature convergence problem that leads to loss of individual diversity in early generations. For the gene improvement the direction (i.e., positive or negative) and the amplitude should be found. The direction can be chosen by comparing the best performance from the previous generation with the current individual performance. The amplitude of the search in [49] is dynamically adapted by diminishing it at each new generation.

The rotation direction is chosen based on a determinant $\mathcal{A}_k = \begin{vmatrix} \alpha_k^b & \alpha_k^j \\ \beta_k^b & \beta_k^j \end{vmatrix}$, where the elements are provided by a qubit from the previous generation best individual $qb_k^b = (\alpha_k^b, \beta_k^b)$ and the current individual that has to be improved $qb_k^j = (\alpha_k^j, \beta_k^j)$ The direction of the rotation angle for improvement toward the best individual is given by $-sign(\mathcal{A}_k)$. The direction is not determined if $\mathcal{A}_k = 0$.

Let V^b be the best individual from the previous generation and V_j an individual of the current generation. If their fitness functions are Jp^b and Jp_j, respectively, the improvement of V_j consists of applying the rotation improvement vector $\Theta = [\theta_0, \theta_1, \ldots, \theta_{g-1}]$. Let θ_{max} and θ_{min} be the maximum and minimum accepted rotation angle, respectively.

If an adaptation of the rotation angle in the generation w of the maximum z generations is used, the function for an individual improvement becomes as in Algorithm 1.

Algorithm 1 Function $Improve(V_j, Jp_j, V^b, Jp^b, \theta_{max}, \theta_{min}, z, w, v)$

```
Require: $V_j, Jp_j, V^b, Jp^b, \theta_{max}, \theta_{min}, z, w, v$
Ensure: $V_j$ is improved
  $\Delta\theta_w \leftarrow \theta_{max} - \frac{\theta_{max}-\theta_{min}}{z} \cdot w \cdot v$
  for all $qb_k \in V_j$ do
    if $(Jp^b > Jp_j) \wedge (\mathcal{A}_k = \begin{vmatrix} \alpha_k^b & \alpha_k^j \\ \beta_k^b & \beta_k^j \end{vmatrix} \neq 0)$ then
      $\theta_k^j \leftarrow -sign(\mathcal{A}_k) \cdot \Delta\theta_w$
    else
      $\theta_k^j \leftarrow (0.5 - random()) \cdot \Delta\theta_w$
    end if
    $qb_k^j \leftarrow A(\theta_k^j) \cdot qb_k^j$
  end for
  return $V_j$
```

The coefficient v was added for convergence adjustment.

The *quantum crossover* operator introduced in [57] was developed and applied in the current research. It works simultaneously on the entire population $\mathcal{P}^w$ of dimension pd (assumed even) at the generation w. It is split into two equal parts, $\mathcal{P}_0^w$ and $\mathcal{P}_1^w$. After ordering P^w according to the chosen fitness function, it is randomly copied in P_0^w and P_1^w. The crossover operator (consisting of swapping the second parts of two individuals' genotypes) is applied simultaneously to all the individuals, and so the population $\mathcal{P}^{w+1}$ is given by Algorithm 2.

Algorithm 2 Function $Crossover(\mathcal{P}^w, pd)$

```
Require: $\mathcal{P}^w, pd$
Ensure: $\mathcal{P}^{w+1}$ is genetically improved
  * parallel $Jp^w \leftarrow simulate(\mathcal{P}^w)$
  $sort(\mathcal{P}^w)$
  * randomly $(\mathcal{P}_0^w, \mathcal{P}_1^w) \leftarrow \mathcal{P}^w$
  for all $V_j \in \mathcal{P}^w$ do
    * swap the last halves of $V_j$ and $V_{pd-1-j}$
  end for
  return $\mathcal{P}^{w+1} \leftarrow \mathcal{P}_0^w \bigcup \mathcal{P}_1^w$
```

The classical mutation operator acts on a genotype, randomly choosing a gene and randomly switching its value. The *quantum mutation operator* acts on the entire population, simultaneously rotating all the qubits with random angles.

2.12.4. Quantum Inspired Genetic Algorithms

QIGA needs a different kind of genome [48] that can be stored in the vector V according to Equation (4). Usually, all the individuals $j; j = 0, 1, \cdots, pd-1$ have the initial genotypes:

$$V_j^0 = [(\varrho, \varrho), (\varrho, \varrho), \cdots, (\varrho, \varrho)]^T \tag{57}$$

The individual genotype modification is performed by a unitary transformation $A_1(\theta)$ (given by (21)) with $\theta = random() \cdot 2\pi$, where $random() \in (0,1)$ is a function that generates random values. The transformation $A_1(\theta)$ in an iteration k acts on a qubit qb_i of V:

$$qb_i^j = A_1(\theta_k) \cdot qb_i^{j-1}. \tag{58}$$

The individual's qubits can be rotated with the same θ or using different rotation angles for each qubit. The individual is obtained by a transformation $\Gamma_j = \mathcal{T}(V_j)$.

The individual assessment is performed by the simulation of the QPN model. Their selection for reproduction is performed by the classical roulette wheel.

Based on Figure 16, the QIGA is constructed by detailing the mappings (as in Algorithm 3). The following notations are used:

- z is the maximum accepted number of generation/iterations,
- w is the current iteration,
- pd the population dimension,
- $\mathcal{P}^w$ is the population in the generation w,
- V^b, Jp^b are the best individual of the previous generation and its fitness function.

Algorithm 3 QIGA

```
Require: $pd, z, \theta_{max}, \theta_{min}, v, V^0, Jp^0$ the initial best individual and its performance
Ensure: $V^b, Jp^b$ the best individual of the last population and its performance
  $\mathcal{P}^0 \leftarrow [V_j^0]_{j=0,1,\ldots,pd-1}$
  $V^b \leftarrow V^0$
  $Jp^b \leftarrow Jp^0$
  $w \leftarrow 0$
  while $(w \leqslant z)$ do
    for all $V_j \in \mathcal{P}^w$ do
      $Jp_j^w \leftarrow simulate(QPN, V_j^w)$ (Figure 17)
    end for
    for all $V_j^w \in \mathcal{P}^w$ do
      $V_j^{w+1} \leftarrow Improve(V_j^w, Jp_j^w, V^b, Jp^b, \theta_{max}, \theta_{min}, z, w, v)$
    end for
    Find $(V^b, Jp^b) \leftarrow \max_{Jp_i; V_i^w \in \mathcal{P}^w} \{(V_j^w, Jp_j^w)\}$
    $w \leftarrow w + 1$
  end while
  return $V^b, Jp^b$
```

The crossover operation can lead the GA search to conserve a part of the path if and where some individuals perform better [59]. This involves the modification of the previous algorithm by replacing *Improvement* with *Crossover* and *Mutation*.

2.12.5. Hybrid Classic Quantum Genetic Algorithms (HCQGAs)

HCQGAs perform the individual improvement implemented in CC and the individual evaluations in QC using an architecture similar to those presented in Figure 15.

The algorithm implemented on CC constructs the population and sends it for the evaluation on QC. QC parallelly and simultaneously evaluates all the individuals and provides the best individual with its score.

The individuals evaluation is performed by partitioning the QDCW positions related to the partitions: T_n, T_g, T_p and T_d corresponding to the case when the positions (i, j) are neutral (i.e., neither on target or trap) zone, on target zone, on trap zone and on degenerated zone if it is on a target and a trap (simultaneously) zone, respectively. The inclusions in these four sets can be coded by two qubits, and the degree of walker inclusion is proposed to be assessed by three qubits. The QDCW simulation, its control (i.e., Γ) and assessment (i.e., Φ) are run simultaneously.

The quantum system evolves in the Hilbert space:

$$\mathcal{H} = \mathcal{H}_\Gamma^4 \bigotimes \mathcal{H}_O^4 \bigotimes \mathcal{H}_X^8 \bigotimes \mathcal{H}_Y^8 \bigotimes \mathcal{H}_J^2 \bigotimes \mathcal{H}_\Phi^{12} \quad (59)$$

where $\mathcal{H}_J^2$ provides the information related to the performance given by the current QDCW position. Similar to the walker state (i.e., orientation and position on X and Y), the individual assessment is given by the position inclusion in an assessment partition and the corresponding degrees.

The HCQGA is composed of two parts: Classic Component of GA and Quantum Component of GA. The algorithms they execute are presented in Algorithms 4 and 5.

Algorithm 4 Classic Component of GA

```
Require: $pd, z, \theta_{max}, w, \theta_{min}, v$
Ensure: $V^b, \Phi^b$ the best individual of population and its performance
  $\mathcal{P}^0 \leftarrow [V_j^0]_{j=0,1,\dots,pd-1}$
  $w \leftarrow 0$
  while $(w \leqslant z)$ do
    set$(\mathcal{P}^w \leftarrow [V_j^w]_{j=0,1,\dots,pd-1})$
    get$(\Phi^w, V^b, \Phi^b)$
    $w \leftarrow w + 1$
    if w<z then
      for all $V_j \in \mathcal{P}^w$ do
        $V_j^w \leftarrow Improve(V_j^{w-1}, \Phi_j^{w-1}, V^b, \Phi^b, \theta_{max}, \theta_{min}, z, w, v)$
      end for
    end if
  end while
  return $V^b, \Phi^b$
```

Algorithm 5 Quantum Component of GA

```
Require: $\mathbf{o}(0), \mathbf{x}(0), \mathbf{y}(0), w, z$ and $QDCW$ model
Ensure: $\Phi^w, V^b, \Phi^b$ the best individual of population and its performance
  $w \leftarrow 0$
  while $(w \leqslant z)$ do
    $\mathcal{P}^w \leftarrow$ get$(\mathcal{P}^w)$
    for all $V_j \in \mathcal{P}^w$ do
      $\Phi_j^w \leftarrow simulate(QDCW, V_j)$
    end for
    $Find\ (V^b, \Phi^b) \leftarrow bubble\_Sorting(V_j, \Phi_j)$
    set$(\Phi^w \leftarrow [\Phi_j]_{j=0,1,\dots,pd-1}, V^b, \Phi^b)$
    $w \leftarrow w + 1$
  end while
```

The methods set and get were used for the description of communication between classic and quantum computers.

For QDCW simulation, another 3 qubits are required for the clepsydra.

The QLC has to be endowed with the necessary quantum gates for the mappings implementation.

The use of the quantum genetic operators *Crossover* and *Mutation* would lead to complete quantum genetic algorithm.

2.12.6. Complete Quantum Genetic Algorithm (CQGA)

This is implemented completely on a quantum computer that executes Algorithm 6.

Algorithm 6 CQGA

```
Require: $pd, z$
Ensure: $V_0$ is the best found solution
  $\mathcal{P}^0 \leftarrow [V_j^0]_{j=0,1,\ldots,pd-1}$;
  $w \leftarrow 0$
  while $(w \leqslant z)$ do
    parallel $Jp^w \leftarrow simulate(QPN)$ (Figure 17)
    $sort(\mathcal{P}^w)$
    $\mathcal{P}^{w+1} \leftarrow Crossover(\mathcal{P}^w, pd)$
    $\mathcal{P}^{w+1} \leftarrow Mutation(\mathcal{P}^{w+1}, pd)$
    $w \leftarrow w + 1$
  end while
  return measure $V_0$
```

3. Results

The QDRW models were tested on a software company simulator, but all the genetic algorithms were tested by simulation using Java language. For QPN, a framework can be found at "https://github.com/dahliajanabi/QPN".

For experimentation of the proposed methods, the previously presented QDRW was modified to the move control of a Quantum Discrete Controlled Walker (QDCW).

3.1. Classic GA Solution of Walker Move Control

The quantum QDCW model is similar to the QPN displayed in Figure 13. The classic walker model has the same graph (Figure 13), but the guards and mappings have to be modified according to Object Enhanced Time Petri Nets (see reference [68] for details). For this case, the transitions t_0, t_1, t_2 and t_3 perform the rotations and the transitions t'_0, t'_1, t'_2 and t'_3 perform the move to the next positions. Unlike the quantum case, the classic walker can be in only one place p_0, p_1, p_2 or p_3, because the superposition is not used in the classic computation.

The classic computation deterministic walker has the model:

$$\begin{bmatrix} o_x(\tau+1) \\ o_y(\tau+1) \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} o_x(\tau) \\ o_y(\tau) \end{bmatrix} \tag{60}$$

$$\begin{bmatrix} x(\tau+1) \\ y(\tau+1) \end{bmatrix} = \begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix} + \begin{bmatrix} o_x(\tau+1) \\ o_y(\tau+1) \end{bmatrix} \tag{61}$$

with $o_x(\tau)$ and $o_y(\tau)$ describing the orientation on axes X and Y, respectively. The initial condition is $[o_x(0), o_y(0), x(0), y(0)]^T$. The control matrix has the argument $\theta = \gamma(\tau) \cdot \pi/2$ with $\gamma(\tau) \in \{0, 1, 2, 3\}$.

The GA genome Γ is a vector of rotations $\gamma(\tau)$ for $0 \leq \tau < 15$.

The usual mutation and crossover genetic operations were performed and the obtained results are presented in Figure 18 for the initial state $[1, 1]$, with orientation $[0, 0]$, $TargetPoint = \{(5, 5)\}$ and $TrapPoints = \{(2, 4), (3, 3), (4, 2)\}$. The best individual genotype is $\Gamma = [2, 0, 0, 0, 1, 0, 0, 0, 0, 0, -, -, -, -, -]$, and the resulting best trajectory is: $(1, 1) * (2, 1) * (3, 1) * (4, 1) * (5, 1) * (5, 2) * (5, 3) * (5, 4) * (5, 5)$.

3.2. Application of QIGA for QDCW

For QDCW model, the mappings assigned to transitions t_0, t_1, t_2 and t_3 (that correspond to generalized coin throw) in the QDRW model have to be replaced with control values applied to qubit registers. The qubit register is composed of two qubits for orientation, three qubits for X and another three qubits for Y.

The probabilities of the best individual obtained by QIGA are represented in Figure 19.

The evolution of the solution search is given in Table 4. The maximum values Ψ^g and Ψ^p were obtained at different moments on time.

Table 4. Search evolution of QIGA.

w	Jp	θ_0^γ	θ_1^γ	θ_0^u	θ_1^u	Ψ^g	Ψ^p
1	5.184191	1.970711	0.172057	5.517499	3.816113	0.327686	0.252499
20	6.151463	5.105799	2.803489	4.184666	5.264229	0.5896	0.173016
50	25.99559	0.098936	0.117976	5.426092	3.997787	0.73656	0.261046
100	26.98992	1.604711	0.136218	5.446864	3.971793	0.883818	0.094816
200	30.26643	1.622495	1.675234	0.00887	1.554262	0.948257	0.296517
400	30.40533	1.50628	1.515821	6.277452	1.575274	0.971144	0.26478

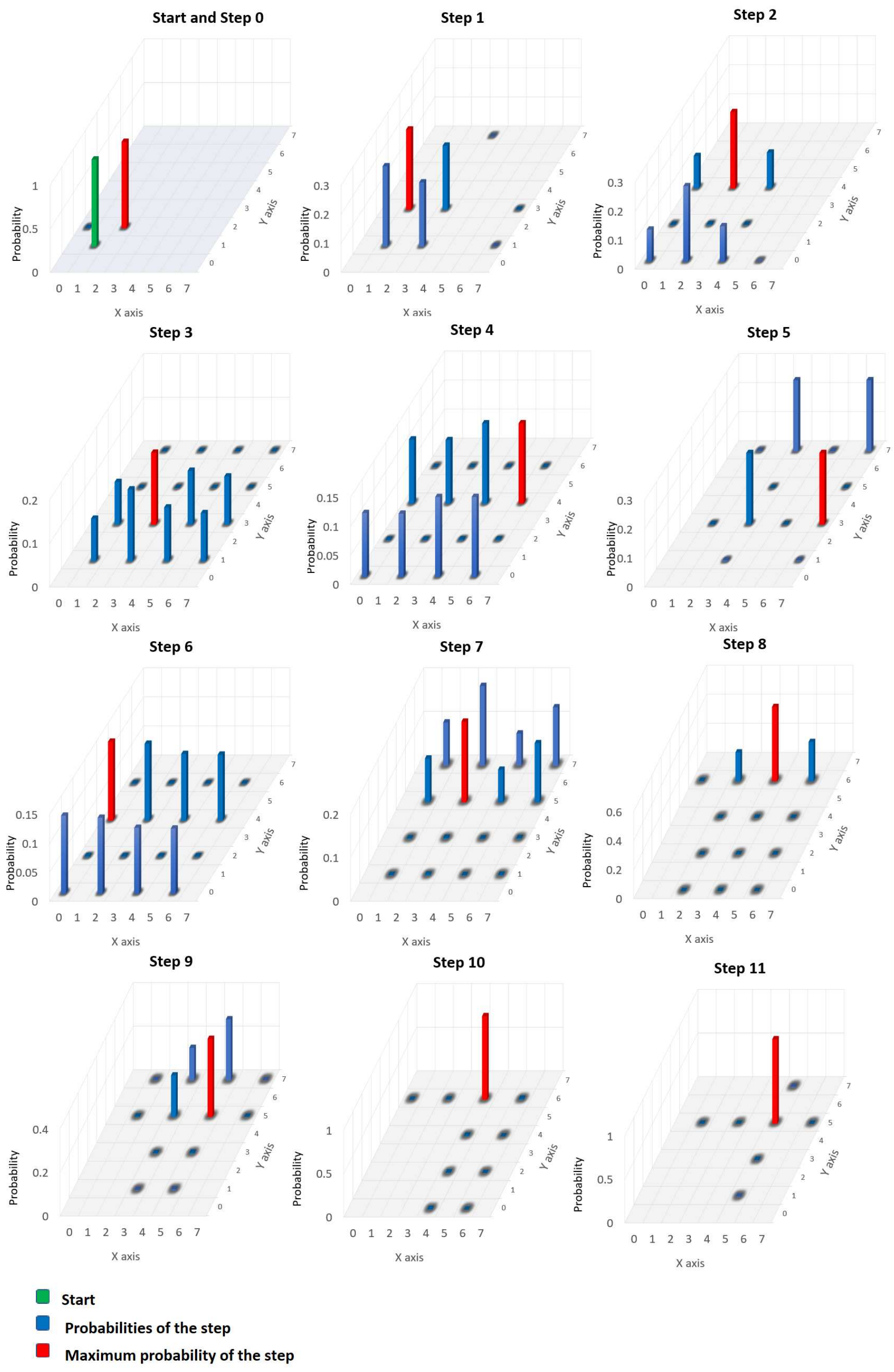

Figure 19. The representation of the probabilities of the best individual for QIGA.

3.3. Application HQCGA for QDCW Control

The values provided by the quantum fitness function where transformed into classic values using the formula:

$$Jp = (\Phi_{10} - 2^{16}) \cdot 10^{-4} \tag{62}$$

where Φ_{10} represents the conversion in the base 10 of Φ.

The probabilities of the best individual obtained by HCQGA are represented in Figure 20.

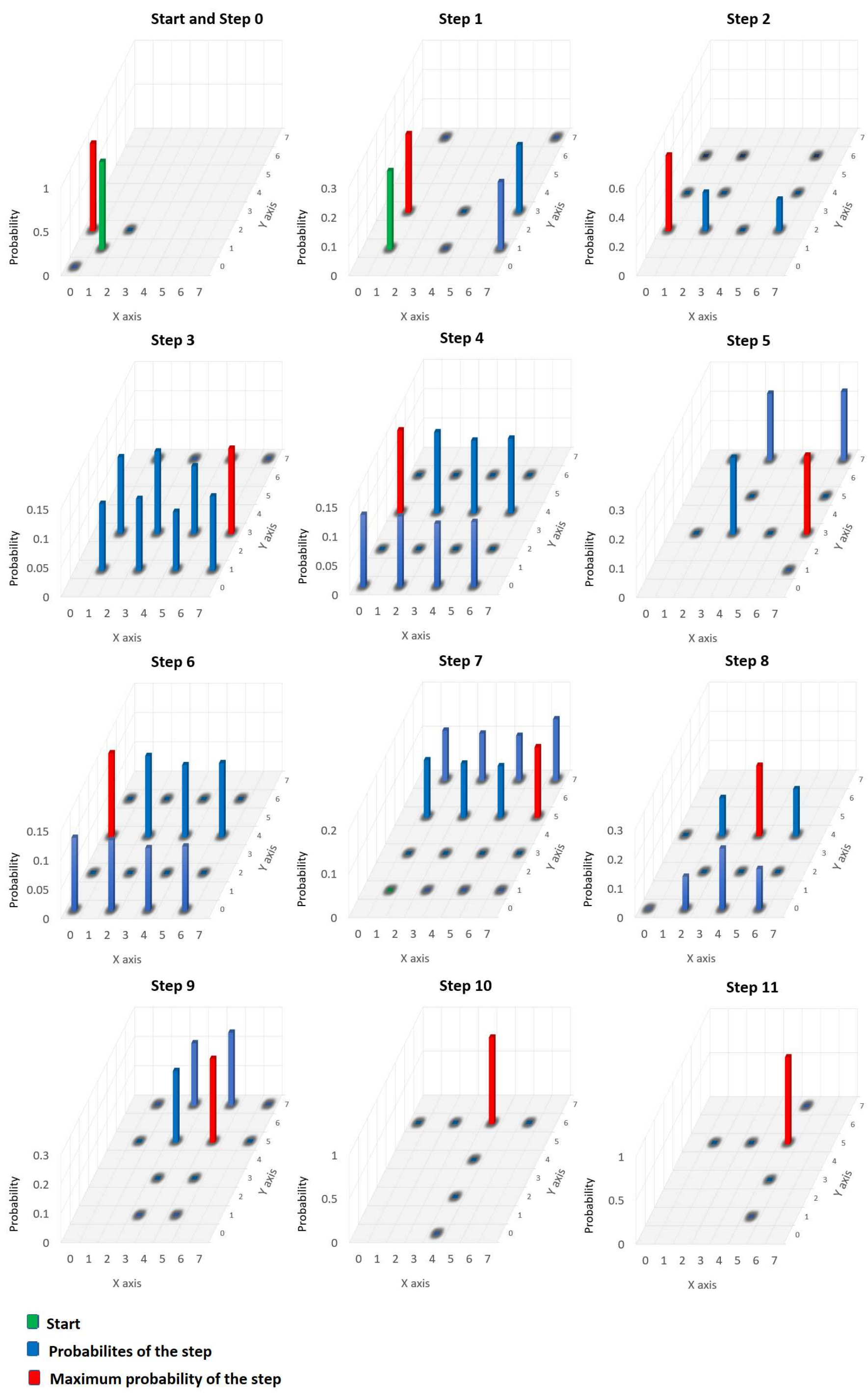

Figure 20. The representation of the probabilities of the best individual for HCQGA.

The evolution of the solution search is given in Table 5.

Table 5. Search evolution of HCQGA.

w	Jp	θ_0^γ	θ_1^γ	θ_0^u	θ_1^u	Ψ^g	Ψ^p
1	−0.2591	4.327966	4.355129	4.691925	5.285935	0.32319	0.249785
20	0.9694	4.69221	0.926491	1.752213	4.315906	0.48180	0.201644
50	1.0063	5.785132	3.212869	3.069224	1.579819	0.59034	0.172353
100	1.1924	2.841270	4.073705	1.163401	2.024726	0.71703	0.205691
200	1.3248	1.533609	1.703517	4.7155721	0.047902	0.91005	0.2772
400	1.3426	1.533609	1.545308	4.715572	6.279499	0.9891	0.261999

3.4. Complete Implemented QGA for QDCW Control

The probabilities of the best individual obtained by CQGA are represented in Figure 21.

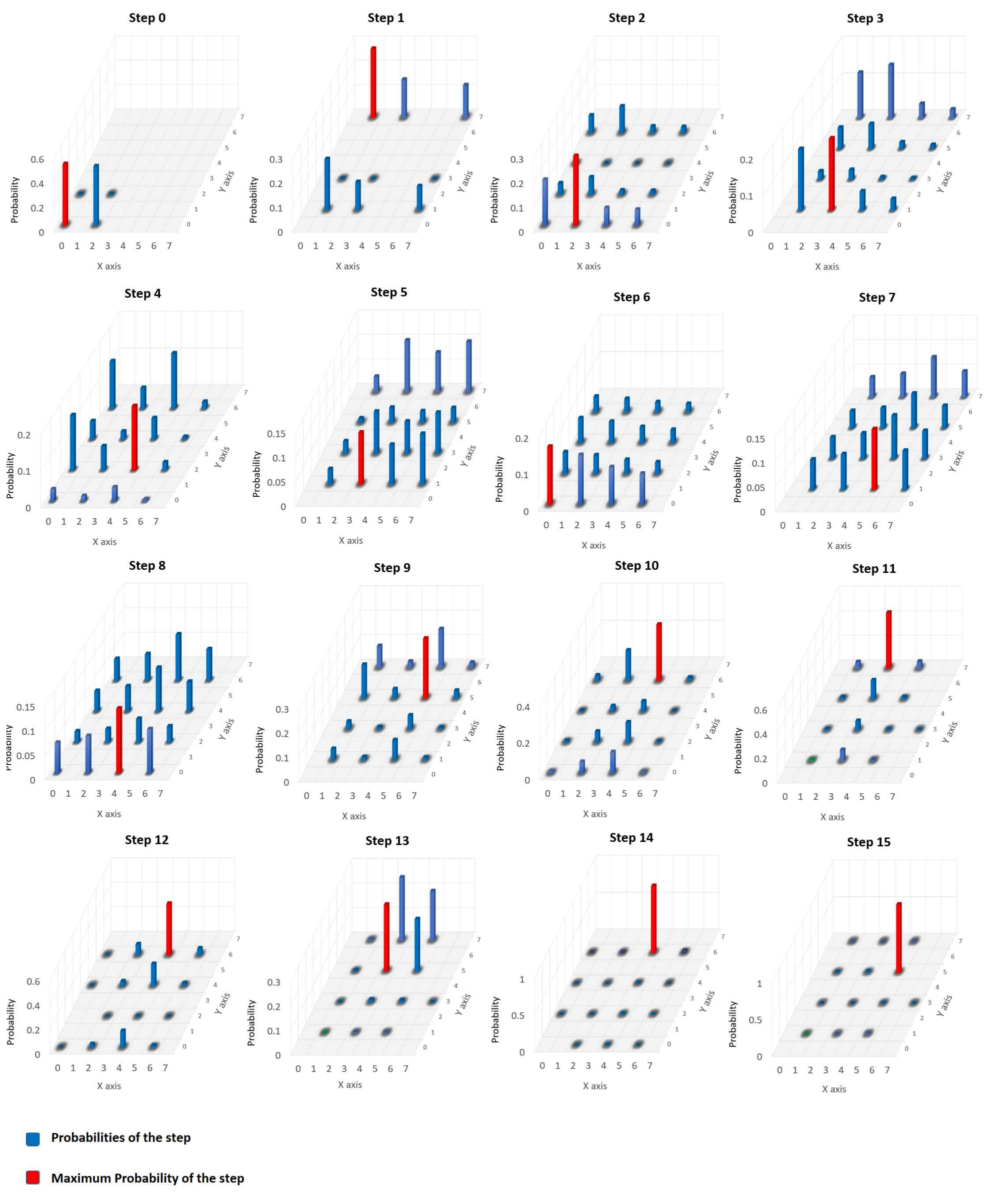

Figure 21. The representation of the probabilities of the best individual for CQGA.

The evolution of the solution search is given in Table 6.

Table 6. Search evolution of CQGA.

w	Jp	θ_0^γ	θ_1^γ	θ_0^u	θ_1^u	Ψ^g	Ψ^p
1	−0.2169	4.718726	1.826285	0.868892	5.209561	0.372534	0.149553
20	1.0427	6.133167	1.175452	4.349668	5.18827	0.560429	0.1686
50	1.2230	6.133167	0.1.142914	4.247011	5.18827	0.749876	0.178494
100	1.2408	6.133167	1.456658	0.868892	2.269625	0.859324	0.18782
200	1.2631	6.133167	1.456658	0.868892	2.25455	0.888608	0.198289
400	1.3220	6.231308	1.515613	0.855429	2.266428	0.930239	0.197465

4. Discussion

Modeling with QPNs can sustain all the phases of quantum software development. They help the understanding of the quantum complex problems and their use facilitate the quantum software conception. QPN can be used for description at the qubit level, or at a higher-level modeling complex mappings. The Petri net-based language can be used for analysis of classic programs as well as quantum programs. Because quantum programs are expected to be used in tandem with classic programs, the Petri nets are useful for constructing a bridge between quantum computers and classic computers, successfully showing their concurrent cooperation. The reachability graph can be used for the classic program and for the quantum program as well. The development based on QPN provides the mappings that lead to the QLC structure required for quantum program implementation. The QPN places show the necessary quantum registers and their dimensions. The transitions have assigned matrices that have to be implemented by QLCs.

The current approach concerns the control of dynamic systems where statistic features are relevant. This kind of applications require fitness functions assessing the temporal behaviors as they were introduced here.

The classic discrete walker has a deterministic behavior, so its optimal control leads to a deterministic solution. QDCW has a stochastic behavior described by superposition and so its optimal control is described by superposition too. Both walkers are controlled by changing their orientations.

The classic controller genome is a time variable vector Θ, while the quantum controller genome is composed of the initial orientation function $\Gamma(0)$ and a matrix U for this function correction.

QIGA and HCQGA use similar individual evolution algorithms (genotypes improvement). Their results are close to each other. CQGA uses quantum crossover and mutation and these lead to a different solution, slightly with lower performance than the previous GAs.

Author Contributions: Conceptualisation, T.S.L.; Experiments design, O.P.C.; Test and verification, E.M.D.-P.; Implementation, D.A.-J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

QC	Quantum Computation or Computer
CC	Classic Computation or Computer
PN	Petri Net
OETPN	Object Enhanced Time Petri Net
QPN	Quantum Petri Net
QDRW	Quantum Discrete Random Walker
ME	Mobile Entity
EA	Evolutinary Algorithm
QEA	Quantum Evolutionary Algorithm
PSO	Particle Swarm Optimization
QME	Quantum Mobile Entity
GA	Genetic Algorithm
CGA	Classic Genetic Algorithm
QGA	Quantum Genetic Algorithm
QIGA	Quantum Inspired Genetic Algorithm
HCQGA	Hybrid Classic Quantum Genetic Algorithm
CQGA	Complete Quantum Genetic Algorithm
QAOA	Quantum Approximate Optimization Algorithm
ES	Evolutionary System
QES	Quantum Evolutionary System

References

1. Bacon, D.; van Dam, W. Recent Progress in Quantum Algorithms. *Commun. ACM* **2010**, *53*, 84–93. [CrossRef]
2. Grumbling, E.; Horowitz, M. (Eds.) *Quantum Computing. Progress and Prospects. A Consensus Study Report of The National Academies of Sciences, Engineering, and Medicine*; The National Academies Press: Washington, DC, USA, 2019. [CrossRef]
3. Letia, T.S.; Durla Pasca, E.M.; Al-Janabi, D.M. Quantum Petri Nets. In Proceedings of the 25th International Conference on System Theory, Control and Computing (ICSTCC), Iasi, Romania, 20–23 October 2021. [CrossRef]
4. Cho, C.-H; Chen, C.-Y.; Chen, K.-C.; Huang, T.-W.; Hsu, M.-C.; Cao, N.-P.; Zeng, B.; Tan, S.-G.; Chang, C.-R. Quantum computation: Algorithms and Applications. *Chin. J. Phys.* **2021**, *72*, 248–269. [CrossRef]
5. Montanaro, A. Quantum algorithms: An overview. *npj Quantum Inf.* **2015**, 2, 15023. [CrossRef]
6. Mavroeidis, V.; Vishi, K.; Zych, M.D.; Jøsang, A. The Impact of Quantum Computing on Present Cryptography. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 3. [CrossRef]
7. Grover, L.K. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **1997**, *79*, 325–328. [CrossRef]
8. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
9. Rossi, M.; Asproni, L.; Caputo, D.; Rossi, S.; Cusinato, A.; Marini, R.; Agosti, A.; Magagnini, M. Using Shor's algorithm on near term Quantum computers: A reduced version. *arXiv* **2021**, arXiv:2112.12647v1.
10. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef] [PubMed]
11. McClean, J.R.; Romero, J.; Babbush, R.; Aspuru-Guzik, A. The theory of variational hybrid quantum-classical algorithms. *arXiv* 2015, arXiv:1509.04279v1. [CrossRef]
12. Jethwani, D.; Le Gall, F.; Singh, S.K. Quantum-Inspired Classical Algorithms for Singular Value Transformation. *arXiv* **2012**, arXiv:1910.05699. [CrossRef]
13. Titiloye, O.; Crispin, A. Quantum annealing of the graph coloring problem. *Discret. Optim.* **2011**, *8*, 376–384. [CrossRef]
14. Apers, S.; Gilyén, A.; Jeffery, S. A Unified Framework of Quantum Walk Search. *arXiv* **2019**, arXiv:1912.04233v1. [CrossRef]
15. Kadian, K.; Garhwal, S.; Kumar, A. Quantum walk and its application domains: A systematic review. *Comput. Sci. Rev.* **2021**, *41*, 100419. [CrossRef]
16. Shenvi, N.; Julia Kempe, J.; Whaley, K.B. Quantum random-walk search algorithm. *Phys. Rev. A* **2003**, *67*, 052307. [CrossRef]
17. Potocek, V.; Gabris, A.T.; Kiss, T.; Jex, I. Optimized quantum random-walk search algorithms on the hypercube. *Phys. Rev. A* **2008**, *79*, 012325. [CrossRef]
18. Godsil, C.; Zhan, H. Discrete-time quantum walks and graph structures. *J. Comb. Theory Ser. A* **2019**, *167*, 181–212. [CrossRef]
19. Portugal, R. *Quantum Walks and Search Algorithms*; Springer: Berlin/Heidelberg, Germany, 2013. [CrossRef]
20. Kendon, V. How to Compute Using Quantum Walks. In Proceedings of the QSQW 2020, EPTCS 315, Marseille, France, 20–24 January 2020; pp. 1–17. [CrossRef]
21. Chawla, P.; Roopesh Mangal, R.; Chandrashekar, C.M. Discrete-time quantum walk algorithm for ranking nodes on a network. *Quantum Inf. Process.* **2020**, *19*, 158. [CrossRef]
22. Li, M.; Liu, C.; Li, K.; Liao, X.; Li, K. Multi-task allocation with an optimized quantum particle swarm method. *Appl. Soft Comput. J.* **2020**, *96*, 106603. [CrossRef]

23. Panahiyan, S.; Fritzsc, S. One-dimensional quantum walks driven by two-entangled-qubit coins. *Phys. Lett. A* **2020**, *384*, 126673. [CrossRef]
24. Zhou, W. Review on Quantum Walk Algorithm. *J. Phys. Conf. Ser.* **2021**, *1748*, 032022. [CrossRef]
25. Ambainis, A. Quantum walk algorithm for element distinctness. *SIAM J. Comput.* **2007**, *37*, 210–239. [CrossRef]
26. Venegas-Andraca, S.E. Quantum walks: A comprehensive review. *Quantum Inf. Process.* **2012**, *11*, 1015–1106. [CrossRef]
27. Choi, J.; Kim, J. A Tutorial on Quantum Approximate Optimization Algorithm (QAOA): Fundamentals and Applications. In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 19–21 October 2019. [CrossRef]
28. Dong, Y.; Meng, X.; Lin, L.; Kosut, R.; Whaley, K.B. Robust Control Optimization for Quantum Approximate Optimization Algorithms. *IFAC PapersOnLine* **2020**, *53*, 242–249. [CrossRef]
29. Bengtsson, A.; Vikstål, P.; Warren, C.; Svensson, M.; Gu, X.; Kockum, A.F.; Krantz, P.; Križan, C.; Shiri, D.; Svensson, I.M.; et al. Improved success probability with greater circuit depth for the quantum approximate optimization algorithm. *Phys. Rev. Appl.* **2020**, *14*, 034010. [CrossRef]
30. Sloss, A.N.; Gustafson, S. 2019 Evolutionary Algorithms Review, Neural and Evolutionary Computing (cs.NE). *arXiv* **2019**, arxiv:1906.0887. [CrossRef]
31. Sofge, D.A. Prospective algorithms for quantum evolutionary computation. In Proceedings of the Second Quantum Interaction Symposium (QI-2008), Saarbrcken, Germany, 22–30 November 2008. [CrossRef]
32. Lahoz-Beltra, R. Quantum Genetic Algorithms for Computer Scientists. *Computers* **2016**, *5*, 24. [CrossRef]
33. Han, K.-H.; Kim, J.-W. Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization. *IEEE Trans. Evol. Comput.* **2002**, *6*, 6. [CrossRef]
34. Patvardhan, C.; Bansal, S.; Srivastav, A. Quantum-Inspired Evolutionary Algorithm for difficult knapsack Problems. *Memetic Comp.* **2015**, *7*, 135–155. [CrossRef]
35. Zhang, R.; Wang, Z.; Zhang, H. Quantum-Inspired Evolutionary Algorithm for Continuous Space Optimization Based on Multiple Chains Encoding Method of Quantum Bits, Hindawi. *Math. Probl. Eng.* **2014**, *2014*, 620325. [CrossRef]
36. Kuo, S.-Y.; Chou, Y.-H. *Entanglement-Enhanced Quantum-Inspired Tabu Search Algorithm for Function Optimization*; IEEE: Piscataway, NJ, USA, 2017. [CrossRef]
37. Konara, D.; Sharma, K.; Sarogi, V.; Bhattacharyya, S. *A Multi-Objective Quantum-Inspired Genetic Algorithm (Mo-QIGA) for Real-Time Tasks Scheduling in Multiprocessor Environment*; Elsevier: Amsterdam, The Netherlands, 2018.
38. Agrawal, R.K.; Kaur, B.; Agarwal, P. Quantum inspired Particle Swarm Optimization with guided exploration for function optimization. *Appl. Soft Comput. J.* **2021**, *102*, 107122. [CrossRef]
39. Zamani, H.; Nadimi-Shahraki, M. H.; Gandomi, A.H. QANA: Quantum-based avian navigation optimizer algorithm. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104314. [CrossRef]
40. Vaze, R.; Deshmukh, N.; Kumar, R.; Saxena, A. Development and application of Quantum Entanglement inspired Particle Swarm Optimization. *Knowl.-Based Syst.* **2021**, *219*, 106859. [CrossRef]
41. Zouache, D.; Abdelaziz, F.B. A cooperative swarm intelligence algorithm based on quantum-inspired and rough sets for feature selection. *Comput. Ind. Eng.* **2017**, *115*, 26–36. [CrossRef]
42. Ganesan, V.; Sobhana, M.; Anuradha , G.; Yellamma, P.; Devi, O.R.; Prakash, K.B.; Naren, J. Quantum inspired meta-heuristic approach for optimization of genetic algorithm. *Comput. Electr. Eng.* **2021**, *94*, 107356. [CrossRef]
43. Dey, S.; Siddhartha Bhattacharyya, S.; Maulik, U. Quantum inspired genetic algorithm and particle swarm optimization using chaotic map model based interference for gray level image thresholding. *Swarm Evol. Comput.* **2014**, *15*, 38–57. [CrossRef]
44. Nezamabadi-pour, H. A quantum-inspired gravitational search algorithm for binary encoded optimization problems. *Eng. Appl. Artif. Intell.* **2015**, *40*, 62–75. [CrossRef]
45. Cao, B.; Fan, S. Zhao, J.; Yangd, P.; Muhammade, K.; Tanveer, T. Quantum-enhanced multiobjective large-scale optimization via parallelism. *Swarm Evol. Comput.* **2020**, *57*, 100697. [CrossRef]
46. Bhatia, M.; Sood, S.K.; Kaurc, S. Quantum-based predictive fog scheduler for IoT applications. *Comput. Ind.* **2019**, *111*, 51–67. [CrossRef]
47. Li, M.; Shang, Y. Generalized exceptional quantum walk search. *New J. Phys.* **2020**, *22*, 123030. [CrossRef]
48. Han, K.-H.; Kim, J.-H. Genetic quantum algorithm and its application to combinatorial optimization problem. In Proceedings of the Congress on Evolutionary Computation, Kraków, Poland, 16–19 July 2000; pp. 1354–1360.
49. Wang, H.; Liu, J.; Zhi, J.; Fu, C. The Improvement of Quantum Genetic Algorithm and Its Application on Function Optimization. *Math. Probl. Eng.* **2013**, *2013*, 730749. [CrossRef]
50. Haipeng, K.; Ni, L.; Yuzhong, S. Adaptive double chain quantum genetic algorithm for constrained optimization problems. *Chin. J. Aeronaut.* **2015**, *28*, 214–228. [CrossRef]
51. Rizk, Y.; Awad, M. A quantum genetic algorithm for pickup and delivery problems with coalition formation. *Procedia Comput. Sci.* **2019**, *159*, 261–270. [CrossRef]
52. Ajagekar, A.; You, F. Quantum computing for energy systems optimization: Challenges and opportunities. *Energy* **2019**, *179*, 76–89. [CrossRef]
53. Hilali-Jaghdam, I.; Ishak, A.B.; Abdel-Khalek, S.; Jamal, A. Quantum and classical genetic algorithms for multilevel segmentation of medical images: A comparative study. *Comput. Commun.* **2020**, *162*, 83–93. [CrossRef]

54. Kaveh, M.; Kamalinejad, H.; Arzani, H. Quantum evolutionary algorithm hybridized with Enhanced colliding bodies for optimization. *Structures* **2020**, *28*, 1479–1501. [CrossRef]
55. Ajagekar, A.; Humble, T.; You, F. Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems. *Comput. Chem. Eng.* **2020**, *132*, 106630. [CrossRef]
56. Wang, L.; Tang, F.; Wu, H. Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation. *Appl. Math. Comput.* **2005**, *171*, 1141–1156. [CrossRef]
57. Ibarrondo, R.; Gatti, G.; Sanz, M. Quantum Algorithm with Individuals in Multiple Registers. *arXiv* **2022**, arXiv:2203.15039v1.
58. Ardelean, S.M.; Udrescu, M. Graph coloring using the reduced quantum genetic algorithm. *PeerJ Comput. Sci.* **2022**, *8*, e836. [CrossRef]
59. Zhang, J.; Kang, M.; Li, X.; Liu, G. Bio-Inspired Genetic Algorithms with Formalized Crossover Operators for Robotic Applications. *Front. Neurorobot.* **2017**, *11*, 56. [CrossRef] [PubMed]
60. Ghosh, M.; Dey, N.; Mitra, D.; Chakrabarti, A. A Novel Quantum Algorithm for Ant Colony Optimization. *IET Res. J.* **2021**, *3*, 13–29. [CrossRef]
61. Zeng, Y.X.; Shen, J.; Hou, S.C.; Gebremariam, T.; Li, C. Quantum control based on machine learning in an open quantum system. *Phys. Lett. A* **2020**, *384*, 126886. [CrossRef]
62. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft. Comput.* **2017**, *22*, 387–408. [CrossRef]
63. Kelly, A. Simulating Quantum Computers Using OpenCL. *arXiv* **2018**, arXiv:1805.00988.
64. Abramsky, S. Petri Nets, Discrete Physics, and Distributed Quantum Computation. In *Concurrency, Graphs and Models*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5065. [CrossRef]
65. Schmidt, H.W. How to Bake Quantum into Your Pet Petri Nets and Have Your Net Theory Too, Computer Science > Software Engineering. *arXiv* **2021**, arxiv:2106.03539.
66. Anres-Martinez, P.; Heunen, C. Weakly measured while loops: Peeking at quantum states. *Quantum Sci. Technol.* **2022**, *7*, 025007. [CrossRef]
67. Kempe, J. Quantum random walks—An introduction overview. *arXiv* **2003**, arxiv:quant-ph/0303081v1.
68. Letia, T.S.; Al- Janabi, D. Object Enhanced Time Petri net models. In Proceedings of the 2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 24–26 May 2018. [CrossRef]
69. Udrescu, M.; Prodan, L.; Vladutiu, M. Implementing Quantum Genetic Algorithms: A Solution Based on Grover's Algorithm. In Proceedings of the 3rd Conference on Computing Frontiers, Ischia, Italy, 3–5 May 2006; pp. 71–82.