information

Article

Quantum-Inspired Evolutionary Algorithm for Optimal Service-Matching Task Assignment

Joan Vendrell * and Solmaz Kia

Department of Mechanical and Aerospace Engineering, University of California Irvine, Irvine, CA 92697, USA
* Correspondence: jvendrel@uci.edu; Tel.: +1-602-585-7492

Abstract: This paper proposes a quantum-inspired evolutionary algorithm (QiEA) to solve an optimal service-matching task-assignment problem. Our proposed algorithm comes with the advantage of generating always feasible population individuals and, thus, eliminating the necessity for a repair step. That is, with respect to other quantum-inspired evolutionary algorithms, our proposed QiEA algorithm presents a new way of collapsing the quantum state that integrates the problem constraints in order to avoid later adjusting operations of the system to make it feasible. This results in lower computations and also faster convergence. We compare our proposed QiEA algorithm with three commonly used benchmark methods: the greedy algorithm, Hungarian method and Simplex, in five different case studies. The results show that the quantum approach presents better scalability and interesting properties that can be used in a wider class of assignment problems where the matching is not perfect.

Keywords: quantum-inspired evolutionary algorithm; service-matching task assignment; greedy algorithm; Hungarian method; Simplex

Citation: Vendrell, J.; Kia, S. Quantum-Inspired Evolutionary Algorithm for Optimal Service-Matching Task Assignment. *Information* **2022**, *13*, 438. https://doi.org/10.3390/info13090438

Academic Editor: Francesco Fontanella

Received: 1 August 2022
Accepted: 9 September 2022
Published: 17 September 2022

1. Introduction

This paper considers a linear optimal service-matching task assignment problem defined by

$$Z^* = \arg\ \min \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{T}} c_{i,j} Z_{i,j} \quad \text{s.t.} \tag{1a}$$

$$Z_{i,j} \in \{0,1\}, \quad i \in \mathcal{S} = \{1,\cdots,N\},\ j \in \mathcal{T} = \{1,\cdots,M\}, \tag{1b}$$

$$\sum_{j\in\mathcal{T}} Z_{i,j} = 1, \quad \forall i \in \mathcal{S}, \tag{1c}$$

$$\sum_{i\in\mathcal{S}} Z_{i,j} \leq 1, \quad \forall j \in \mathcal{T}, \tag{1d}$$

where $Z = [Z_{i,j}]$ is the assignment matrix with N rows and M columns. We refer to $\mathcal{S}$ as *service agent set* and its elements as *service agent*, and to $\mathcal{T}$ as *task set* and its elements as *task*. Thus, $c_{i,j} > 0$ is the cost of the assignment of service agent $i \in \mathcal{S}$ to task $j \in \mathcal{T}$, and $Z_{i,j} = 1$ means that service agent $i \in \mathcal{S}$ is assigned to task $j \in \mathcal{T}$. Without loss of generality, we assume that $N \leq M$, i.e., the task set, is larger than the service agent set. The imposed constraint (1c) ensures that each service agent gets only one task, while constrain (1d) limits the assignment of any task to, at most, one service agent. Z^* is the adjacency matrix of the resultant matching graph after the assignment is peformed.

An example scenario inspired by the service-matching mobile-agent deployment for coverage over a dense set of targets studied in [1] is shown in Figure 1. In this service-matching problem, service UAVs A1, A2, A3 (flying at the same altitude), whose spatial service over the horizontal two-dimensional (2D) plane is described by a Gaussian distribution, should be deployed to provide coverage to a set of dense targets in a 2D flat plane. The

targets are clustered into subgroups TC1, TC2, TC3, and TC4, each described by a Gaussian distribution of the targets. The assignment cost $c_{i,j}$ is the measure of similarity between the service distribution of agent Ai and the distribution of targets in subgroup TCj, measured in terms of the Kullback–Leibler divergence (KLD) ([2], p. 34). More details are provided in Section 4, where we conduct a numerical study of this service-matching task assignment.

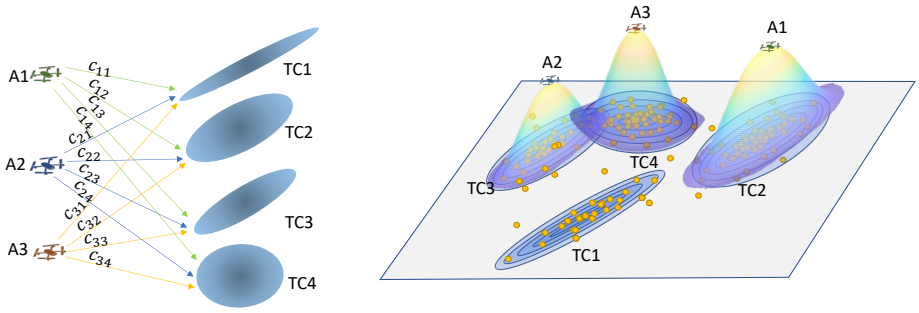

Figure 1. An optimal service-matching UAV deployment via AP formulation. $c_{i,j}$ indicates the level of similarity between agent Ai's service distribution with the target cluster TCj's distribution. The figure depicts a case that agent A1, A3, and A3 are assigned, respectively, to TC2, TC3 and TC4.

There are many other service-matching applications which can be used with the optimal AP problem (1). For example, in manufacturing, in assembly lines, where there must be a planning between the operators and the tasks [3,4]. Other situations such as in delivery tasks or ridesharing [5] can also be formulated in the same way. In addition, some social, economical or political situations are no more than an AP [6].

As in other combinatorial optimisation problems, there is a lack of methods that can achieve a good approximation factor in polynomial time with good scalability. Currently, the main algorithms still used to solve service-matching problems (1) are classical methods such as the greedy algorithm, which, theoretically, ensures an approximation factor of 63% [7] with a complexity of $\mathcal{O}(NM \log M)$ [8]. The Hungarian method, with an approximation factor of 1 and a complexity of $\mathcal{O}(NM(N+M))$ [8], or the Simplex method with, also, an approximation factor of 1 and a complexity of $\mathcal{O}(N^2M^2)$ [8]. The reader should note that the aforementioned complexity bounds for the Hungarian and Simplex algorithms are based on execution of these algorithms when $N = M$.

With the rise of machine learning, new techniques such as neural networks, reinforcement learning or evolutionary algorithms have been formulated to solve combinatorial optimisation problems. Related to all these techniques, the advances in quantum computation have originated a new field of study: quantum-inspired algorithms. Quantum-inspired algorithms try to formulate classical algorithms in a quantum way, using qbits and its properties, in order to increase the speed of these methods (Quantum-computing- related terminologies used here are defined in Section 2. The interested reader can find more detailed information in [9], Chapter 1.) For example, in [10], they use the superposition property in order to accelerate distribution shaping in the travelling salesman problem, or in [11], they use entanglement to prevent a two-agent autonomous system from cyber-physical attacks. In [12] is seen how quantum formulation helps in clustering problems and in [13,14] is seen how quantum properties are interesting for obtaining a better balance between exploration and exploitation in reinforcement learning. The same idea of reducing model size and increasing robustness is seen in other deep-learning techniques in [15]. One of the interesting applications of quantum-inspired algorithm development is explored in [16], where a quantum-inspired version of an evolutionary algorithm (EA) is implemented to solve a knaspack problem with a proven improvement with respect to classical EA.

This paper focuses on developing a computationally efficient suboptimal solution for the service-matching assignment problem (1) using a quantum-inspired EA (QiEA) algorithm and demonstrates its application in solving a service-matching coverage problem

illustrated in Figure 1. Special attention is given to the design of the collapsing procedure to make it more coherent with the real collapsing process of qbits and to speed up the process. In Section 2, preliminary notions about quantum theory are presented. The proposed method is presented in Section 3. In Section 4, a numerical study focused on a service-matching task assignment demonstrates the efficacy of our proposed algorithm. The conclusions and our plan for future work are explained in Section 5.

2. Preliminaries

This section gives the definition of the quantum formulation notion and terminologies we use throughout the paper. For the convenience of the reader, we also provide the definition of the KLD measure.

First, following [17], we introduce the notion and the definitions from the quantum formulation. In quantum computation, the basic unit of information is called a *Qbit* and it is neither one nor zero, but the superposition of both states. That means that, in one timestamp, there is a certain probability that the qbit is in in state zero or in state one. This can be formulated using Dirac's notation, also known as brakets, as,

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

where $|\psi\rangle$ is the qbit, $|0\rangle$ and $|1\rangle$ are the two possible states and α and $\beta \in \mathbb{C}$ are the amplitudes of these states. The Born's rule tells us that these amplitudes are directly related to the probability of being in state $|0\rangle$ or $|1\rangle$ as the probability of one state is the power of its amplitude,

$$p(\langle 0|\psi\rangle) = |\,\langle 0|\psi\rangle\,|^2 = |\alpha\,\langle 0|0\rangle + \beta\,\langle 0|1\rangle\,|^2 = |\alpha|^2, \tag{3}$$

$$p(\langle 1|\psi\rangle) = |\,\langle 1|\psi\rangle\,|^2 = |\alpha\,\langle 1|0\rangle + \beta\,\langle 1|1\rangle\,|^2 = |\beta|^2, \tag{4}$$

$$|\alpha|^2 + |\beta|^2 = 1, \tag{5}$$

One qbit remains in this superposition until it is measured or observed. When that happens, the qbit becomes fixed in one of the states in the function of the amplitudes, this process is called *collapsing*.

As explained in Equation (2), in the superposition state, a qbit depends on two parameters, $\alpha = a_\alpha + \mathrm{j}\,b_\alpha$ and $\beta = a_\beta + \mathrm{j}\,b_\beta$, that are indeed complex numbers. Therefore, qbits can be represented with spherical coordinates,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = (a_\alpha + \mathrm{j}\,b_\alpha)\,|0\rangle + (a_\beta + \mathrm{j}\,b_\beta)\,|1\rangle = r_\alpha \mathrm{e}^{\mathrm{j}\,\varphi_\alpha}\,|0\rangle + r_\beta \mathrm{e}^{\mathrm{j}\,\varphi_\beta}\,|1\rangle \tag{6}$$

applying the Born's rule, Equation (5),

$$|\psi\rangle = \mathrm{e}^{-\mathrm{j}\,\varphi_\alpha}\left(\cos(\frac{\theta}{2})\,|0\rangle + \sin(\frac{\theta}{2})\,\mathrm{e}^{\mathrm{j}\,(\varphi_\beta - \varphi_\alpha)}\,|1\rangle\right) \tag{7}$$

were $0 \leq \theta \leq \pi$ and $0 \leq \varphi \leq 2\pi$. It can be demonstrated that $\mathrm{e}^{-\mathrm{j}\varphi_\alpha}$ has no observable effects [18]; therefore, effectively, we only have to consider the following equation,

$$|\psi\rangle = \cos(\frac{\theta}{2})\,|0\rangle + \mathrm{e}^{\mathrm{j}\phi}\sin(\frac{\theta}{2})\,|1\rangle \tag{8}$$

From Equation (8), it can be seen that a qbit belongs to a Hilbert space $\mathcal{H}$ and can be represented as a sphere. This representation is known as a Bloch Sphere, Figure 2.

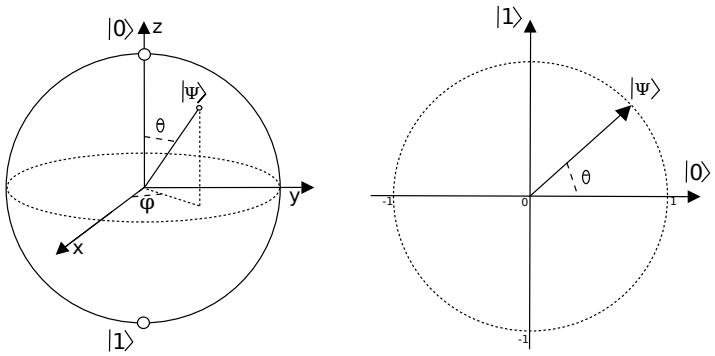

Figure 2. Representation of a qbit: On the **left**, the Bloch Sphere, a spherical representation of a qbit. On the **right**, a circular representation of a qbit after all the assumptions.

From the Bloch Sphere, it can be seen that the state of a qbit can be modified using rotations. Indeed, in quantum computation, the logic gates used to develop circuits are no more than rotation operations. In this work, the only important rotation is the θ rotation, as it is the one that makes the superposition state closer or further to states $|0\rangle$ or $|1\rangle$. Therefore, in this case, qbits can be represented as circles and, in order to modify them, a simple rotation matrix will be used.

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{9}$$

KLD is a measure of similarity (dissimilarity) between two probability distributions $p(x)$ and $q(x)$, where the smaller the value the more similar two distributions are. KLD is zero if and only if the two distribution are identical ([2], p.34). For Gaussian distributions, $p(x) = \mathcal{N}(\mu_0, \Sigma_0)$ and $q(x) = \mathcal{N}(\mu_1, \Sigma_1)$, the KLD has a closed form expression ([19], Equation (2))

$$\mathrm{D}_{\mathrm{KL}}\Big(p(x)||q(x)\Big) = \frac{1}{2}\Big(\ln\frac{|\Sigma_1|}{|\Sigma_0|} + (\mu_0-\mu_1)^\top\Sigma_1^{-1}(\mu_0-\mu_1) + \top(\Sigma_1^{-1}\Sigma_0) - n\Big), \tag{10}$$

where n is the dimension of the distributions.

In what follows, we rely on a function best defined as

$$\hat{Z} = \mathsf{best}(Z^1, Z^2, \cdots, Z^p), \tag{11}$$

which, given any finite number of assignment matrices $Z^1, \cdots, Z^p$, return the assignment matrix $\hat{Z} \in \{Z^1, Z^2, \cdots, Z^p\}$ that results in the smallest of cost $\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{T}} c_{i,j}Z^l_{i,j}$, $l \in \{1, \cdots, p\}$, computed according to the assignment cost (1a).

3. QiEA for Optimal Service Matching

Inspired by the concept of quantum computing of qbit representation, rotation, observation and qbit collapsing, we propose the QiEA Algorithm 1 as a suboptimal solution to the optimal service-matching problem (1). To design this algorithm, we proceeded as follows.

Consider the assignment matrix $Z = [Z_{i,j}]$. For any given service agent $i \in \mathcal{S}$, every feasible assignment consists of assigning 1 to one of the elements of $Z_{i,j}$, $j \in \mathcal{T}$ and zero to the rest, while respecting also (1d). Since we do not know which element should be 1, the idea is to start by a probabilistic approach and, for every service agent $i \in \mathcal{S}$, attach a notion of probability to every $Z_{i,j}$, which indicates with what probability that element is 1. Then, at each iteration t of the algorithm, this probability is adjusted as we describe below. The notion of probability for each element $Z_{i,j}$ can be realised using quantum qbits,

$|\phi_{i,j}\rangle\,(t) = \alpha_{i,j}(t) + \mathrm{j}\,\beta_{i,j}(t)$. Thus, at each step t of the algorithm, for each individual l in population set $\mathcal{P} = \{1, \cdots, K\}$, there is a qbit matrix

$$|\Phi\rangle^{l}(t) = \begin{bmatrix} \begin{bmatrix} \alpha^{l}_{1,1}(t) & \alpha^{l}_{1,2}(t) & \cdots & \alpha^{l}_{1,M}(t) \\ \beta^{l}_{1,1}(t) & \beta^{l}_{1,2}(t) & \cdots & \beta^{l}_{1,M}(t) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \alpha^{l}_{N,1}(t) & \alpha^{l}_{N,2}(t) & \cdots & \alpha^{l}_{N,M}(t) \\ \beta^{l}_{N,1}(t) & \beta^{l}_{N,2}(t) & \cdots & \beta^{l}_{N,M}(t) \end{bmatrix} \end{bmatrix} \tag{12}$$

with size $2N \times M$ where each element (i, j) is the mentioned probability $|\phi_{i,j}\rangle^{l}(t)$ of the assignment between agent $i \in \mathcal{S}$ and task $j \in \mathcal{T}$.

Algorithm 1 QiEA algorithm for service-matching assignment

procedure QIEA
  Input size of population K, number of iteration MAX_GEN
  $t \leftarrow 0$
  initialise $|\Phi\rangle^{1}(t), \cdots, |\Phi\rangle^{K}(t)$
  generate K individuals $Z^1(t) = [Z^1_{i,j}(t)], \cdots, Z^K(t) = [Z^K_{i,j}(t)]$ by collapsing $|\Phi\rangle^{l}(t)$
  for $l \in \{1, \cdots, K\}$ **do**
    $B^{l}_{\text{local}}(t) \leftarrow$ evaluate $Z^{l}(t) = [Z^{l}_{i,j}(t)]$
  end for
  $B(t) \leftarrow \text{best}(B^{1}_{\text{local}}(t), \cdots, B^{K}_{\text{local}}(t))$
  while $t \leq$MAX_GEN **do**
    $t \leftarrow t + 1$
    for $l = \{1, \cdots, K\}$ **do**
      generate $Z^{l}(t) = [Z^{l}_{i,j}(t)]$ by collapsing $|\Phi\rangle^{l}(t-1)$
      $B^{l}_{\text{local}}(t) \leftarrow$ evaluate $Z^{l}(t) = [Z^{l}_{i,j}(t)]$
      update $|\Phi\rangle^{l}(t)$
      Update $B^{l}_{\text{local}}(t-1)$ if $B^{l}_{\text{local}}(t)$ is better
      if migration condition **then**
        $B^{l}_{\text{local}}(t) \leftarrow B(t)$
      end if
    end for
  end while
  $\hat{Z}^{\star} \leftarrow \text{best}(B^{1}_{\text{local}}(\text{MAX\_GEN}), \cdots, B^{K}_{\text{local}}(\text{MAX\_GEN}), B(\text{MAX\_GEN}))$
end procedure

Evolutionary algorithms are based on the Darwinian explanation of species evolution where, step by step, there are mutations in each individual of the species and, by the end, only the individuals with better adaptation to the new environment survive. The evolutionary algorithms follow a two-step procedure [20]. First, we generate an initial population of individuals randomly. Next, we repeat the following re-generational steps until termination: (a) the fitness of each individual in the population is evaluated (lowest cost, feasibility), (b) the fittest individuals are selected as parents for reproduction, (c) new individuals are bred (from parents) through crossover and mutation operations to give birth to offspring, (d) the least-fit individuals of the population are replaced with new individuals.

QiEA is a probabilistic algorithm similar to other evolutionary algorithms. We use an evolutionary method to go from an initial state of uncertainty $|\Phi\rangle^{l}(0)$ to a final determined system $\hat{Z}^{*}$, which is a suboptimal solution to (1). The underlying concept in QiEA is that $|\Phi\rangle^{l}(t)$ informs us on how to select individual $Z(t) = [Z_{i,j}(t)]$ at each step of the algorithm (an individual is selected by collapsing $|\Phi\rangle^{l}(t)$). $|\Phi\rangle^{l}(t)$, which determines the probability

of each element of $Z(t) = [Z_{i,j}(t)]$ being 0 or 1, is updated at each step based on the fitness of the generated individuals at step $t-1$ using the rotation method, which will be described below. The advantage of QiEA is, indeed, in this step of updating the probabilities of selecting future individuals based on feedback from the fitness of the generated individuals. The schematic procedure of our QiEA algorithm is shown in Figure 3. Our proposed QiEA is shown in Figure 3: there is a K parallel process to generate K individuals that constitutes the population set of the algorithm. Each individual generation process $l \in \mathcal{P} = \{1, \cdots, K\}$ has its own $|\Phi\rangle^l(t)$, initialised at $|\Phi\rangle^l(0)$ and updated at each iteration of the algorithm.

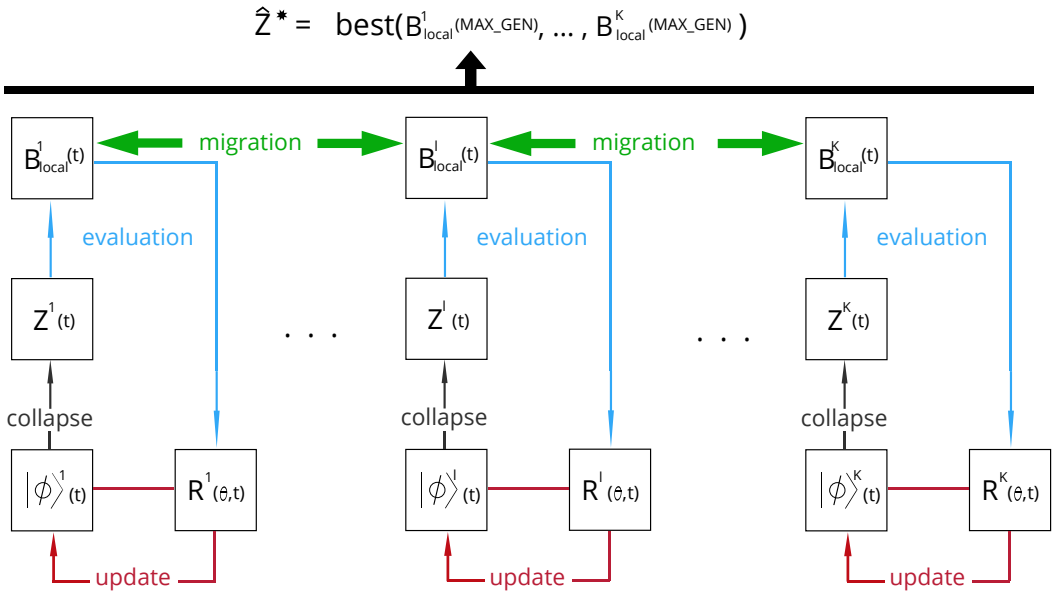

Figure 3. Scheme of the QiEA procedure considering K initial populations.

Our proposed QiEA algorithm consists of three main procedural steps, *collapsing*, *evaluation and updating* and *migration*.

Collapsing procedure: This process is a novel contribution of this work and it is a faithful representation of the quantum-like collapsing of the qbit systems. With this purpose, the collapsing is performed using the maximum randomness possible. At each step t to generate individual $Z^l(t)$, first, one random service i and one random target j are selected to form the agent–task pair (i, j). Then, with a uniform probability, we generate a random value $\gamma^l_{i,j}(t) \in [0, 1]$. Then, $Z^l_{i,j}(t)$ is generated using the collapsing procedure below

$$Z^l_{i,j}(t) = \begin{cases} 1 & \text{if } \gamma^l_{i,j}(t) \leq |\beta^l_{i,j}(t)|^2 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

After collapse, the agent–task pair (i, j) is removed from the random selection, and the process is repeated until there is no service agent left to be selected (recall our assumption that there is more service agents than the tasks). Our collapsing procedure is summarised in Algorithm 2. It is important to note that, if a service agent is assigned to one task (qbit (i, j) is collapsed to one), no other service agent can be assigned to that task and that service agent can not perform any other task. That is exactly what we formulated in constraints (1c) and (1d). That is, our individual generation procedure always generates feasible individuals $Z^l(t)$, and does not need the costly and often ad-hoc *repairing* of the individuals that other quantum-inspired algorithms, e.g., [10,21,22], need. The elimination of the repair step also results in faster convergence.

Algorithm 2 Collapsing

procedure COLLAPSING
  Input Qbit system $|\Phi\rangle^l(t)$
  $\bar{\mathcal{S}} \leftarrow [1, \cdots, N]$
  $\bar{\mathcal{T}} \leftarrow [1, \cdots, M]$
  while $\bar{\mathcal{S}} \neq \{\}$ **do**
    select random service $i \in \bar{\mathcal{S}}$
    select random service $i \in \bar{\mathcal{T}}$
    generate $\gamma^l_{i,j}(t)$ uniformly from $[0, 1]$
    collapse: $Z^l_{i,j}(t) \xleftarrow{\text{Equation (13)}} (|\phi\rangle^l_{i,j}(t), \gamma^l_{i,j}(t))$
    if $Z^l_{i,j}(t) = 1$ **then**
      remove i from $\bar{\mathcal{S}}$
      remove j from $\bar{\mathcal{T}}$
    end if
  end while
end procedure

Evaluation procedure: This procedure has two parts, the computation of the cost for fitness test, and the formulation/updating of the rotation matrix based on the feedback from the fitness analysis. Given the collapsed individuals $Z^1(t), \cdots, Z^K(t)$, at step t, firstly, the cost of the assignment $f(Z^l(t)) = \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{T}} c_{i,j} Z^l_{i,j}(t)$ is computed. The fitness evaluation not only depends on the cost at time t due to $Z^l(t)$ but also has temporal information about the evolution of the system by saving the best obtained solution until that iteration, which is stored as $B^l_{\text{local}}(t)$. The variable B^l_{local} for each individual $l \in \{1, \cdots, K\}$ is initialised by collapsing $|\Phi\rangle^l(0)$, as described in step 5 of Algorithm 1. The fitness analysis relies on function (11) and updates each local B^l_{local} according to $B^l_{\text{local}}(t) = \text{best}(Z^l(t), B^l_{\text{local}}(t-1))$. Next, in the evaluation procedure, we update the qubit matrix $|\Phi\rangle^l$ according to

$$\begin{bmatrix} \alpha^l_{i,j}(t) \\ \beta^l_{i,j}(t) \end{bmatrix} = R(\theta^l_{i,j}, t) \begin{bmatrix} \alpha^l_{i,j}(t-1) \\ \beta^l_{i,j}(t-1) \end{bmatrix}, \tag{14}$$

where rotation angle $\theta^l_{i,j} \in \{0, \delta\pi, -\delta\pi\}$ is determined from Table 1. Here, $0 < \delta\pi < \frac{\pi}{2}$ is a design parameter which normally is a very small angle. Recall that, according to the circular qbit representation (8), which is visualised in Figure 2, a positive rotation $0 < \delta\pi < \frac{\pi}{2}$ results in increasing the probability of collapsing to 1 and a negative rotation $0 < -\delta\pi < \frac{\pi}{2}$ results in increasing the probability of collapsing to 0. We proposed Table 1 based on the logic that if the value of $Z^l_{i,j}(t)$ and $B^l_{\text{local}_{i,j}}(t-1)$ are the same, we make no changes to the qbit corresponding to the agent–task pair (i, j), thus setting $\theta^l_{i,j}(t) = 0$ (cases 1, 2, 7 and 8 in Table 1). If changing from $B^l_{\text{local}_{i,j}}(t-1) = 1$ to $Z^l_{i,j}(t) = 0$ results in decreasing the total cost, we want to encourage collapsing to $Z^l_{i,j}(t) = 0$ by increasing this probability by using a positive rotation $\theta^l_{i,j} = \delta\pi$ (case 3 in Table 1). On the other hand, if changing from $B^l_{\text{local}_{i,j}}(t-1) = 0$ to $Z^l_{i,j}(t) = 1$ results in decreasing the total cost, we want to encourage collapsing to $Z^l_{i,j}(t) = 1$ by increasing such probability by using a negative rotation $\theta^l_{i,j} = -\delta\pi$ (case 5 in Table 1). If the changes in the values result in increasing the total cost, we do not make any changes to $|\Phi\rangle^l_{i,j}$, i.e., $\theta^l_{i,j}(t) = 0$ (cases 4 and 6 in Table 1). To summarise, we make an adjustment to $|\Phi\rangle^l_{i,j}$ if $Z^l_{i,j}(t) \neq B^l_{\text{local}_{i,j}}(t-1)$ and the best value at step t is set to $Z^l(t)$; thus, we can interpret the change in $B^l_{\text{local}_{i,j}}(t-1)$ value as a contributor to decreasing the cost. Algorithm 3 summarises our evaluation procedure.

Table 1. Evaluation cases to define rotation angle.

Case	$Z^l_{i,j}(t)$	$B^l_{\text{local}_{i,j}}(t-1)$	$f(Z^l(t)) \geq f(B^l_{\text{local}}(t-1))$	$\theta_{i,j}$
1	0	0	false	0
2	0	0	true	0
3	0	1	false	$\delta\pi$
4	0	1	true	0
5	1	0	false	$-\delta\pi$
6	1	0	true	0
7	1	1	false	0
8	1	1	true	0

Algorithm 3 Evaluation

procedure EVALUATION
  Input collapsed system $Z^l(t)$, best saved solution $B^l_{\text{local}}(t-1)$
  $B^l_{\text{local}}(t) = \text{best}(Z^l(t), B^l_{\text{local}}(t-1))$
  $\theta^l_{i,j} = 0$ for any $i \in \mathcal{S}$ and $j \in \mathcal{T}$
  if $B^l_{\text{local}}(t) = Z^l(t)$ **then**
    for $i \in \mathcal{S}$ **do**
      for $j \in \mathcal{T}$ **do**
        if $Z^l_{i,j}(t) < B^l_{\text{local}_{i,j}}(t-1)$ **then**
          $\theta_{i,j} = \delta\pi$
        end if
        if $Z^l_{i,j}(t) > B^l_{\text{local}_{i,j}}(t-1)$ **then**
          $\theta_{i,j} = -\delta\pi$
        end if
      end for
    end for
  end if
  for $i \in \mathcal{S}$ **do**
    for $j \in \mathcal{T}$ **do**
      update $|\Phi\rangle^l_{i,j}(t)$ according to (14)
    end for
  end for
end procedure

Migration procedure: Like other evolutionary algorithms, to avoid getting stuck in local maxima or minima we perform a migration procedure when the *migration condition* $(t - t_{\text{last migration}}) \geq n \times \text{MAX\_GEN}$, where n is a percentage that defines when the process is completed. In the migration step, we simply replace the current best local value $B^l_{\text{local}}(t)$ of all the individuals $l \in \{1, \cdots, K\}$ with $B(t) = \text{best}(B^1_{\text{local}}(t), \cdots, B^K_{\text{local}}(t))$, i.e., $B^l_{\text{local}}(t) = B(t))$ for all $l \in \{1, \cdots, K\}$.

4. Numerical Demonstration

To demonstrate the efficacy of our proposed QiEA, consider the service-matching robot-deployment example we described in Section 1 and depicted in Figure 1. Recall that, in this example, we have a set $\mathcal{S}$ of N service agents with spatial service distribution $p_i(x)$, $i \in \mathcal{S}$ with Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$. Each target cluster TCj, $j \in \mathcal{T}$, extracted from a GMM model of the entire targets distribution, has a Gaussian distribution $q_j(x) \sim \mathcal{N}(\bar{\mu}_j, \bar{\Sigma}_j)$. According to [1], Theorm 2, the best value for $c_{i,j} = \text{D}_{\text{KL}}\left(p_i(x)||q_j(x)\right)$ is obtained when the robot is moved such that $\mu_i = \bar{\mu}_j$ and the principle axes of the two uncertainty ellipsoids

corresponding to Σ_i and $\bar{\Sigma}_j$ are parallel. Considering this configuration, for each service agent we compute $c_{i,j} = \mathrm{D}_{\mathrm{KL}}\left(p_i(x)||q_j(x)\right)$ to form the weighting matrix $[c_{i,j}]$. Next, through a set of numerical simulations, we evaluate the efficacy of our QiEA algorithm in solving this optimal service-matching deployment problem.

In this work, all the simulation codes are developed with Python using well-known open-source libraries. The optimal deployment for service-matching data is randomly generated. We carry out a simulation study that compares the performance of our proposed Algorithm 1 to that of three well-known algorithms: the greedy algorithm, the Hungarian method and the Simplex linear-programming method. The greedy algorithm is an iterative method that greedily selects the smallest value of $c_{i,j}$ to set $Z_{i,j} = 1$, eliminates the row and the column corresponding to (i, j) from the weight matrix $[c_{i,j}]$ and repeats the procedure. The Hungarian method is a similar technique to the Greedy Algorithm but, instead of picking direct assignments from a static cost matrix, the cost matrix is modified at each step in the function of the minimum values of each row and column. Therefore, this method considers not only the minimum cost for each assignment, but also the rest of the assignments, which is why it can achieve the optimal solution. To have a comprehensive study, we also compare our result to that of the Simplex method, which solves the continuous relaxation of the optimal service-matching problem (1).

4.1. Simulation Results

This section presents the results of our numerical experiments. The proposed algorithm is compared to the benchmark methods in five different scenarios. The simulations are conducted via Python programming on a PC with processor Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz and a RAM of 16 GB.

4.1.1. Case Studies

We simulate five different scenarios, which are described in Table 2. Case 1 is the perfect-matching case, where $N = M$. However, there can also be unbalanced cases where the number of target clusters is bigger than the number of the service agents. To demonstrate how our algorithm scales with respect to N and M and also their relative size, we consider various scenarios for when $N < M$. The normal distributions for each target/service in each experiment are randomly created using Python.

Table 2. Case studies.

Case	Number of Service Agents (N)	Number of Target Clusters (M)
1	10	10
2	5	8
3	2	8
4	90	100
5	2	100

4.1.2. Hyper-Parameters of the QiEA Algorithm 1

To execute the QiEA Algorithm 1, the set of hyper-parameters listed in the first column from the left of Table 3 should be selected. We chose three candidate values for each hyper-parameter as reported in Table 3. To tune the hyper-parameters for each case study listed in Table 2, a grid search over 81 different combinations was implemented. During this grid search, it was observed that there can be two groups of hyperparameters: the ones that make the algorithm more accurate and the ones that make the algorithm faster (compared over same number of epochs). As such, we report the simulation results for two different executions of Algorithm 1 for solving our problem of interest for the cases listed in Table 2: the accurate quantum-inspired evolutionary algorithm (AQiEA) and the fast quantum-inspired evolutionary algorithm (FQiEA). Table 4 gives the chosen hyper-

parameters for each case study. Notice that the hyper-parameters are not necessarily the same for each case study.

Table 3. Hyper-parameters of and the assigned possible values for the grid search.

Hyperparameters	Possible Values
Population size (K)	2, 5, 10
Rotation angle ($\delta\pi$)	0.0025π, 0.01π, 0.05π
Epochs (MAX_GEN)	10, 20, 50
Migration frequency	0.25, 0.5, 0.75

Table 4. Grid search results.

Case	Type	Population Size (K)	Rotation Angle ($\delta\pi$)	Epochs (MAX_GEN)	Migration Frequency
1	Accurate	2	0.05π	20	0.5
	Fast	2	0.0025π	20	0.5
2	Accurate	2	0.0025π	50	0.5
	Fast	2	0.01π	50	0.75
3	Accurate	2	0.0025π	10	0.5
	Fast	2	0.01π	50	0.75
4	Accurate	10	0.0025π	20	0.75
	Fast	2	0.01π	10	0.75
5	Accurate	2	0.05π	20	0.25
	Fast	2	0.0025π	50	0.75

4.1.3. Metrics

In order to compare and evaluate the performance of the methods, three metrics have been considered, which include *optimality gap*, *execution time complexity* and a *quality metric*. The optimality gap, denoted by α, is defined as

$$\alpha = \frac{\text{cost computed by the algorithm} - \text{optimal cost}}{\text{optimal cost}}. \tag{15}$$

When $N = M$, the Hungarian and the Simplex methods are guaranteed to produce an optimal solution; however, this guarantee is not the case when $N < M$. To obtain the optimal value for $N < M$, a suggested practice is adding $M - N$ virtual service agents whose assignment costs to task clusters are much greater than the actual service agents'. This way, a perfect matching problem is created, which we use to generate the optimal cost value for computing the optimality gap. Notice that adding virtual agents increases the computational complexity of the algorithms. The simulation results reported below for all the algorithms consider the actual matching cases when no virtual agent is added.

The execution time, measured in milliseconds (ms), is the time it took to run the algorithm. To unify both prior metrics and for the sake of the comparison, the quality metric Q was implemented in order to have a faster and easier way of comparing and discussing the algorithms. This metric is computed as

$$Q = \frac{100 - \alpha}{\text{execution time}}. \tag{16}$$

4.1.4. Simulation Results

The simulation results for the three aforementioned performance metrics are reported in Tables 5–7. Per definition (15), the best optimality result is when $\alpha = 0$. Thus, for a suboptimal solution, we desire an α closer to zero. For Case 1, as expected, the results in Table 5 show that the Hungarian and Simplex methods have zero optimality gap. Notice that in Case 1, AQiEA outperforms the greedy method in the optimality gap, while FQiEA delivers a close performance to that of the greedy algorithm. In the rest of the cases, the

Hungarian and the Simplex methods also deliver a zero optimality gap, but as Table 6 shows, this comes with a high execution time, especially when the scale of the system grows. In Case 4, the Simplex algorithm failed to converge in a reasonable time, so no results are listed for this case in Tables 5–7. Per definition (16), the quality metric Q is designed to be 100 when the optimality gap is zero and the execution time is 1 ms. As the results in Table 7 show, both QiEA algorithms outperform the rest of the methods in terms of the quality metric.

Table 5. Optimality gap α for the case studies.

Case	Greedy	Hungarian	Simplex	AQiEA	FQiEA
1	0.4059	0	0	0.2150	0.4579
2	0.2336	0	0	0.1825	1.135
3	0	0	0	0	4.910
4	0.6298	0	–	0.7622	0.9125
5	0.6365	0	0	2.0178	3.007

Table 6. Execution time in ms for the case studies.

Case	Greedy	Hungarian	Simplex	AQiEA	FQiEA
1	10.03	11.05	537.9	4.591	3.866
2	5.000	7.970	112.5	2.910	2.464
3	4.521	3.000	26.62	2.393	1.530
4	552.9	544.9	-	226.7	32.61
5	19.04	27.90	781.6	1.845	1.418

Table 7. Quality metric for the case studies.

Case	Greedy	Hungarian	Simplex	AQiEA	FQiEA
1	9.930	9.050	0.1859	21.73	25.75
2	19.95	12.55	0.8889	34.30	40.12
3	22.12	33.33	3.757	41.79	62.15
4	0.1797	0.1835	–	0.4377	3.039
5	5.252	3.584	0.1279	52.57	31.73

5. Conclusions

This work showed how a quantum-inspired evolutionary algorithm could be used as a practical solution method to solve combinatorial optimisation problems such as an optimal service-matching coverage problem. Our numerical studies showed that the quantum approach presents better scalability and interesting properties that can be used in a wider class of assignment problems where the matching is not perfect. Our future work will apply our proposed quantum-inspired evolutionary algorithm framework in solving submodular maximisation problems, particularly the optimal welfare problem. We will also study the automated tuning of the algorithm hyper-parameters using a machine-learning approach.

Author Contributions: Conceptualization, J.V. and S.K; methodology, J.V.; software, J.V.; validation, S.K.; resources, S.K.; investigation, J.V. and S.K; writing—original draft preparation, J.V.; writing—review and editing, S.K.; supervision, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a Balsells Fellowship grant and there was no APC.

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:

QiEA	Quantum-inspired evolutionary algorithm
AQiEA	Accurate quantum-inspired evolutionary algorithm
FQiEA	Fast quantum-inspired evolutionary algorithm
EA	Evolutionary algorithm
AP	Assignment problem
KLD	Kullback–Leibler divergence
TC	Target cluster
UAV	Unmanned aerial vehicle

References

1. Chung, Y.; Kia, S.S. A distributed service-matching coverage via heterogeneous agents. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4400–4407.
2. MacKay, D. *Information Theory, Inference and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003; p. 34.
3. Baybars, I. A Survey of exact algorithms for the simple assembly line balancing problem. *Manag. Sci.* **1986**, *32*, 909–932
4. Sivasankaran, P.; Shahabudeen, P. Literature review of assembly line balancing problems. *Int. J. Adv. Technol.* **2014**, *73*, 1665–1694.
5. Simonetto, A.; Monteil, J.; Gambella, C. Real-time city-scale ridesharing via linear assignment problems. *Transp. Res. Part C Emerg. Technol.* **2019**, *101*, 208–232.
6. Siebert, H. The Assignment Problem. In *Beiträge zur mikro- und zur makroökonomik*; Berninghaus, S.K., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 1, pp. 439–448.
7. Sviridenko, M. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.* **2004**, *32*, 41–43.
8. Williamson, D.P.; Shmoys, D.B. *The Design of Approximation Algorithms*; Cambridge University Press: Cambridge, UK, 2011.
9. Woit, P. Introduction. In *Quantum Theory, Groups and Representations: An Introduction*; Department of Mathematics, Columbia University; New York, NY, USA, 2022; pp. 1–12.
10. Soloviev, V.P.; Bielza, C.; Larranaga, P. Quantum-inspired estimation Of distribution algorithm to solve the travelling salesman problem. In Proceedings of the IEEE Congress on Evolutionary Computation, Kraków, Poland, 28 June–1 July 2021; pp. 416–425.
11. Khoshnoud, F.; de Silva, C.W.; Ibrahim, I. Quantum entanglement of autonomous vehicles for cyber-physical security. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Banff, AB, Canada, 5–8 October 2017; pp. 2655–2660.
12. Xiao, J.; Yan, Y.; Zhang, J.; Tang, Y. A quantum-inspired genetic algorithm for k-means clustering. *Expert Syst. Appl.* **2010**, *37*, 4966–4973.
13. Li, Y.; Aghvami, A.H.; Dong, D. Intelligent trajectory planning in UAV-mounted wireless networks: A quantum-inspired reinforcement learning perspective. *IEEE Wirel. Commun. Lett.* **2021**, *10*, 1994–1998.
14. Chen, C.; Yang, P.; Zhou, X.; Dong, D. A Quantum-inspired Q-learning algorithm for indoor robot navigation. In Proceedings of the IEEE International Conference on Networking, Sensing and Control, Sanya, China, 6–8 April 2008; pp. 1599–1603.
15. Rajesh, V.; Parameshwar, U.; Mohana, N. Quantum Convolutional neural networks (QCNN) using deep learning for computer vision applications. In Proceedings of the International Conference on Recent Trends on Electronics, Information, Communication & Technology, Bangalore, India, 27–28 August 2021; pp. 728–734.
16. Han, K.; Kim, J. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans. Evol. Comput.* **2002**, *6*, 580–593
17. Combarro, E.F. *A Practical Introduction to Quantum Computing: From Qubits to Quantum Machine Learning and Beyond*; CERN OpenLab: Geneva, Switzerland, 2020.
18. Messiah, A. *Quantum Mechanics*; Dover Publications Inc.: Mineola, NY, USA, 1999; Volume 1, p. 296.
19. Hershey, J.R.; Olsen, P.A. Approximating the Kullback Leibler divergence between Gaussian mixture models. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Honolulu, HI, USA, 15–20 April 2007; Volume 4, pp. 317–320.
20. Streichert, F. Introduction to evolutionary algorithms. In *Evolutionary Computation 1*; CRC Press: Boca Raton, FL, USA; University of Tübingen: Tübingen, Germany, 2018; pp. 97–101.
21. Chmiel, W.; Kwiecień, J.Quantum-inspired Evolutionary Approach for the quadratic assignment problem. *Entorpy* **2022**, *20*, 781.
22. Zhang, G. Quantum-inspired evolutionary algorithms: A survey and empirical study. *J. Heuristics* **2011**, *17*, 303–351