Soft Computing
https://doi.org/10.1007/s00500-022-06946-8

FOCUS

Quantum OPTICS and deep self-learning on swarm intelligence algorithms for Covid-19 emergency transportation

Habiba Drias[1] · Yassine Drias[2] · Naila Aziza Houacine[1] · Lydia Sonia Bendimerad[1] · Djaafar Zouache[3] · Ilyes Khennak[1]

Accepted: 16 February 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
In this paper, the quantum technology is exploited to empower the OPTICS unsupervised learning algorithm, which is a density-based clustering algorithm with numerous applications in the real world. We design an algorithm called Quantum Ordering Points To Identify the Clustering Structure (QOPTICS) and demonstrate that its computational complexity outperforms that of its classical counterpart. On the other hand, we propose a Deep self-learning approach for modeling the improvement of two Swarm Intelligence Algorithms, namely Artificial Orca Algorithm (AOA) and Elephant Herding Optimization (EHO) in order to improve their effectiveness. The deep self-learning approach is based on two well-known dynamic mutation operators, namely Cauchy mutation operator and Gaussian mutation operator. And in order to improve the efficiency of these algorithms, they are hybridized with QOPTICS and executed on just one cluster it yields. This way, both effectiveness and efficiency are handled. To evaluate the proposed approaches, an intelligent application is developed to manage the dispatching of emergency vehicles in a large geographic region and in the context of Covid-19 crisis in order to avoid an important loss in human lives. A theoretical model is designed to describe the issue mathematically. Extensive experiments are then performed to validate the mathematical model and evaluate the performance of the proposed deep self-learning algorithms. Comparison with a state-of-the-art technique shows a significant positive impact of hybridizing Quantum Machine Learning (QML) with Deep Self Learning (DSL) on solving the Covid-19 EMS transportation.

Keywords Quantum machine learning · Quantum ordering points to identify the clustering structure · Deep self learning AOA · Deep self learning EHO · Emergency transportation

1 Introduction

Quantum mechanics is the theoretical basis of the emerging field of quantum computing with the aim of speeding-up calculations. Recent quantum applications are numerous. For instance, quantum computing could enable accelerating diagnoses in medicine and solve problems in finance. Quantum algorithms of different disciplines such as neural networks and optimization are developing fast. At the same time, industrial firms are working to build and make available quantum computers (Zahorodko 2021). Quantum machine learning (QML) is a new research axis in artificial intelligence field (Bharti et al. 2020; Biamonte 2017; Wittek 2014). It let speed

Communicated by Oscar Castillo.

✉ Habiba Drias
habiba.drias@usthb.edu.dz

Yassine Drias
y.drias@univ-alger.dz

Naila Aziza Houacine
nhouacine@usthb.dz

Lydia Sonia Bendimerad
lbendimerad@usthb.dz

Djaafar Zouache
djaafarzouache@yahoo.fr

Ilyes Khennak
ikhennak@usthb.dz

[1] LRIA, USTHB, BP 32 El Alia Bab Ezzouar, Algiers 16111, Algeria

[2] LRIA, University of Algiers, 02 rue Didouche Mourad, Algiers 16000, Algeria

[3] LRIA, University of Bordj Bou Arréridj, El-Anasser, Bordj Bou Arréridj 34030, Algeria

Published online: 08 April 2022

up classical machine learning algorithms that deal with big data.

Quantum mechanics is a rich collection of powerful theories that provide description of nature. Concepts such as superposition of states, interference, entanglement are expected to provide an exponential speed up to solve complex scientific and industrial problems. As QML has a great attentiveness nowadays, our interest in this paper focuses on developing a quantum version of a density-based clustering algorithm, knowing that the latter has practical applications in numerous domains.

Density-based clustering techniques discover clusters with arbitrary shape in a spatial database. The yielded clusters are dense regions of objects in the data space and the objects outside the clusters are of low density and hence represent noise. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996) is a nice algorithm, very popular in geographic information systems (GIS). In one scan, it clusters objects given two parameters representing, respectively, the maximum distance of the neighborhood from one object and a number of points representing the minimum desired density in an elementary cluster. This algorithm is known to be very sensitive to such parameters, which usually are unknown et consequently hard to determine, especially for real-world and high-dimensional data sets. Intensive experiments should be performed to tune their values in order to achieve the right clustering. The Ordering Points To Identify the Clustering Structure (OPTICS) (Mihael et al. 1999) which is based on similar principles, has the advantage to handle a broad range of parameter settings. It can be seen as an extension from DBSCAN. It is a smart algorithm for automatic and interactive cluster analysis, allowing the ordering of clusters through an ordered file that can be graphically viewed. The cluster ordering allows to extract clustering information such as cluster centers, arbitrary-shaped clusters and the clustering structure.

On the other hand, up-to-date and topical technologies of artificial intelligence are emerging and exploited to solve hard complex problems. Deep learning methods provide a great hope to solve more effectively and efficiently these issues.

We proposed four main contributions in this paper. The first one is a quantum version of OPTICS algorithm, aiming at gaining efficiency for clustering spatial data. The algorithm is called Quantum OPTICS (QOPTICS) and its computational complexity is evaluated and compared to that of the classic counterpart.

The second one consists of a design of deep learning of two swarm intelligence algorithms, the Artificial Orcas Algorithm (AOA) (Bendimerad and Drias 2020) and the Elephant Herding Optimisation [EHO] (Wang et al. 2015), in order to achieve effectiveness for problem solving. Two mutation operators known as Cauchy and Gaussian operators are introduced in the algorithms for that aim. The designed algorithms are called Deep Self Learning AOA (DSLAOA) and Deep Self Learning EHO (DSLEHO).

The third is a hybrid system combining DSLAOA and DSLEHO respectively with QOPTICS in order to gain more efficiency for problem solving. Indeed, running DSLAOA and DSLEHO on just one cluster that is, on a small part of the dataset is reducing calculation time.

The fourth is an application to emergency transportation in the context of COVID-19, which helps managing hospitals during crises of this pandemic. In order to optimize the time transportation of a patient to a health center, QOPTICS is used to cluster hospitals in regions, each one taking in consideration a certain number of emergency calls, knowing that the most important objective for the decision makers in EMS is to save human lives. For this purpose, responding to emergency calls by sending ambulances in the shortest time should be decided quickly. Assigning vehicles to the nearest calls location may cause some regions to become less covered than others and hence calls from these zones will wait a long time to be served. Ambulance dispatching and zones covering problems are managed jointly in order to optimize not only the vehicles arriving time to an emergency call but also the covering of all the calls. DSLAOA and DSLEHO attempt to respond to this goal.

The originality of the study relies on the fact that the tools used in each phase of the proposed system are from recent emerging technologies such as quantum machine learning and deep learning.

This paper is organized in six sections. The next one presents backgrounds on the OPTICS algorithm, the basic quantum algorithms and the dynamic ambulance dispatching and emergency calls covering. The third section describes the proposed quantum OPTICS algorithm alongside with its complexity. The fourth section enchains with the presentation of the deep self-learning method on AOA and on EHO. The fifth describes the adaptation of DSLAOA and DSLEHO to the EMS transportation issue. The sixth section exhibits the experimental results obtained by QOPTICS, which consists in clustering geographical zones containing a great density of hospitals in the Kingdom of Saudi Arabia (KSA) as well as emergency Covid-19 calls providing from this country. The outcomes generated by deep self-learning methods are also presented and discussed. The article ends with a conclusion highlighting the major contributions and future open questions.

2 Background

This section outlines the basic concepts used in the construction of our proposals. First, the OPTICS clustering algorithm is described and discussed. Second, the quantum subroutines

that helped to design QOPTICS are presented. Third, the dynamic ambulance dispatching and emergency calls covering problem is exposed as it serves as an application of the developed algorithms.

2.1 OPTICS clustering algorithm

Clustering n objects of an unordered set $O = \{O_1, O_2, \ldots, O_n\}$ consists in grouping the similar objects in the same cluster and the dissimilar ones in different clusters. To measure the similarity between objects, a distance function is defined. Ordering Points To Identify the Clustering Structure (OPTICS) (Mihael et al. 1999) is an unsupervised density-based clustering algorithm. It can be seen as an improved version of DBSCAN with additional advantages. It computes an ordering of the objects, while associating with each object its core-distance and its reachability distance. This ordering allows the extraction of all clusterings with respect to any distance smaller than the distance parameter. The computational complexity of OPTICS is $O(n^2)$ (Mihael et al. 1999).

2.2 Quantum operators and subroutines

Quantum computing uses quantum mechanics as a theoretical basis, where the concepts of superposition of states, density matrices representing states and the entanglement phenomenon are fundamental. A state in quantum mechanics encompasses statistical information about its position in a n-dimensional Hilbert space. Based on the Dirac notation, it is represented by a vector as in Eq. 1.

$$|\Psi >= \sum_i \alpha_i |b_i > \tag{1}$$

α_i for all i are complex numbers representing the amplitudes of the basis states $|b_i >$ under the normalization condition $\sum_i {\alpha_i}^2 = 1$. The linear combination of all the basis states is called *superposition*. It denotes the fact that the states exist simultaneously all of them in the superposition.

A qubit (quantum bit) is a quantum state that represents the smallest unit of quantum information storage. It consists of a superposition $\alpha_1|0 > +\alpha_2|1 >$ of two basis states that are $|0 >$ and $|1 >$ such that ${\alpha_1}^2 + {\alpha_2}^2 = 1$. The main advantage of the quantum computer is that its computing power is exponential in terms of the number of qubits. In fact, two combined qubits are in a superposition of four states $\alpha_1|00 > +\alpha_2|01 > +\alpha_3|10 > +\alpha_4|11 >$, with ${\alpha_1}^2 + {\alpha_2}^2 + {\alpha_3}^2 + {\alpha_4}^2 = 1$. With n qubits, a superposition of 2^n states can be created. So when an operator is applied to the set of qubits, it is applied to 2^n states at the same time, which is equivalent to a parallel calculation on 2^n data.

One important operation in quantum computing is the measurement, which allows to get a state component with a probability proportional to its weight. For instance, when measuring the value of the qubit, the only answers that can be obtained are 0 or 1. The probability of measuring state 0 is α_1^2, while that of measuring state 1 is equal to α_2^2. After measurement, the qubit is in the measured state in classical computing.

Except the measurement operation, all the quantum operators are unitary and are represented by gates. A quantum algorithm is then designed as a classical algorithm using Toffoli gates involving in some places quantum operators or subroutines in order to speed up the algorithm calculation.

A quantum state is entangled if it cannot be expressed by a combination of all its basis states. Another important concept is the quantum register, which belongs to the Hilbert space with 2^n states and n qubits. Using the Dirac notation, it is represented as $|\Psi >= \sum_{i=0}^{i=2^n-1} \alpha_i |i >$ under the normalization condition $\sum_i {\alpha_i}^2 = 1$. The basis state $|i >$ expresses the binary encoding of i.

2.2.1 Quantum operators

Quantum computers use quantum circuits composed of quantum gates or operators. Among quantum operators is the Toffoli gate that can perform all the operations of classical circuits. The n-bit Toffoli gate is a generalization of the Toffoli gate. It has n input bits $(x_1, x_2, \ldots, x_n)$ and n output bits. The first $n-1$ output bits are unchanged whereas the last one is $(x_1$ AND …AND $x_{n-1})$ XOR x_n. The Hadamard gate transforms the pure state $|0 >$ into the superposed state $\frac{1}{\sqrt{2}}|0 > +\frac{1}{\sqrt{2}}|1 >$ and the pure state $|1 >$ into the superposed state $\frac{1}{\sqrt{2}}|0 > -\frac{1}{\sqrt{2}}|1 >$. The measurement in this case will have the same probability of giving 1 or 0. The application of a Hadamard gate to each qubit of an n-qubit register in parallel is equivalent to the Hadamard transform H_n.

Another important and extensively used gate is the Grover diffusion operator as it has the ability of searching for an element in an unordered list, in just one quantum operation.

2.2.2 Quantum subroutines

Numerous basic quantum subroutines have been developed and published in the recent literature. Those that are used for devising the quantum OPTICS algorithm are described below.

The Grover's algorithm (Grover 1996) is a quantum subroutine that searches for an element x_0 belonging to an unordered set S of n elements. Using a classical algorithm, the best complexity to perform such search is $O(n)$, as in the worst case all the elements will be tested before encounter-

ing x_0. The Grover's algorithm performs the same task within $O(\sqrt{n})$ time complexity. The algorithm considers a function $F : \{0, 1, 2, \ldots, n-1\} \rightarrow \{0, 1\}$ that returns 1 for x_0 and 0 for all the other elements, that is for only x_0, $F(x_0) = 1$. It uses Hadamard gates to create a uniform superposition of all the states at the beginning and then the Grover operator.

In Boyer et al. (1998), the authors provide a generalization of Grover's algorithm, which is a new technique for counting the number of occurrences of x_0 in the list, with the same computational complexity $O(\sqrt{n})$.

Quantum amplitude amplification algorithm (Brassard et al. 2002) is another generalization of Grover's algorithm. Used as a subroutine, it helps to quadratic speedup over several classical algorithms. The quantum amplitude amplification algorithm does not start in a uniform superposition but only initializes states, when information is available on the states. If the probability to find an element a of S is p_a (and not $\frac{1}{n}$ as in Grover's algorithm) and the sum of probabilities of all elements is $\sum_1^{|S|} p_i$, the computational complexity is $O(\frac{1}{\sqrt{p_a}})$.

Using the quantum exponential searching algorithm (Boyer et al. 1998), the authors in Durr and Hoyer (1996) devised a quantum algorithm that yields the minimum of an unordered set with a computational complexity $O(\sqrt{n})$.

In Durr et al. (2006), Durr et al. developed a quant_find_smallest_values subroutine for finding the c closest neighbors of a point in $O(\sqrt{c \times n})$ time.

Although its numerous advanced developments, QML remains rich in open questions. Quantum OPTICS or QOPTICS studied in this paper is one of them.

2.3 The dynamic ambulance dispatching and emergency calls covering

The dynamic ambulance dispatching and emergency calls covering (DADECC-COVID19) system that we propose is described and depicted in Fig. 1. It considers a set of ambulance stations geographically dispersed, a set of hospitals and a set of real time incoming emergency calls. It receives emergency calls and controls the movement of all the vehicles from the station towards the call point and then from this point to the selected hospital. More precisely, when an emergency call arrives, the system makes a decision about the vehicle to send to assist this call. To respond to calls in an optimal time, the system should guarantee a good covering rate of all the zones from where the calls can come. A vehicle can be idle when it is at station, on route towards a call where it can be deviated to serve another call, arriving to a call location and finishing service to become available.

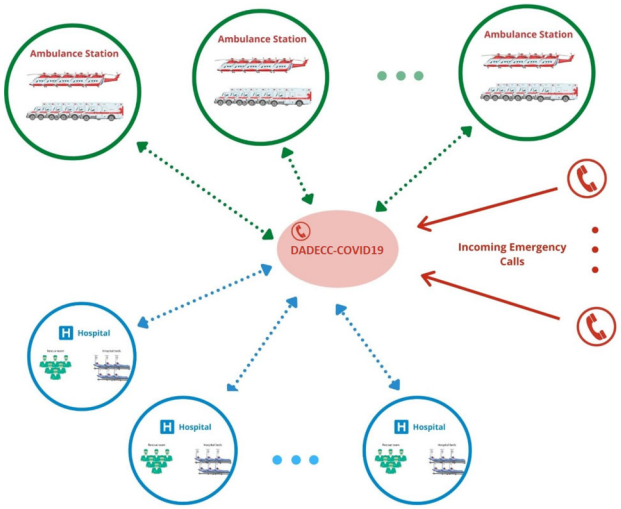


Fig. 1 Dynamic ambulance dispatching and emergency calls covering system (DADECC-Covid19)

The real-time emergency vehicle dispatching problem in this work is dynamic. The destination of vehicles for example can change at any time if their diversion improves the service quality, especially the response time. To avoid deviating the same vehicle many times or deviating too much vehicles with the aim to preserve the emergency crews and the caller from disturbances, a limit is imposed by accepting only reassignments that bring in an important saving of time or calls covering. The proposed solving methods are based on new AI technologies and adapt to this dynamism. Figure 2 illustrates the chain of treatments of an emergency call by DADECC-COVID19.

In the context of COVID-19, numerous studies have been published, the aim being to explore in a way or another some aspects of this unpredictable pandemic. Artificial intelligence has been largely explored in this context. Castillo and his co-author worked on the forecast for the countries based on the COVID-19 time series of confirmed cases and deaths (Castillo and Melin 2020). They also developed a method for a COVID-19 classification of countries based on an intelligent fuzzy fractal approach (Castillo and Melin 2021). For the case of Mexico, an application for predicting the COVID-19 time series was also undertaken (Melin 2020). In this interesting study, a multiple ensemble neural network model with fuzzy response aggregation is proposed. In Mansour (2021), the authors propose an unsupervised deep learning model based on image recognition to detect and classify COVID-19. A preprocessing technique is introduced to enhance the image quality for the diagnosis and another one for the classification.

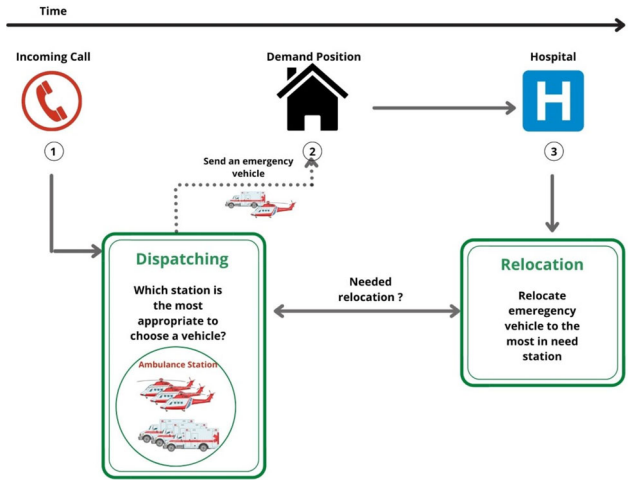


Fig. 2 Emergency call response chain

The present study is conducted for KSA but with another concern, which is managing the emergency transportation during COVID-19 crises.

3 Quantum ordering points to identify the clustering structure

The QOPTICS algorithm we devise stores its input data in a quantum random access memory (QRAM) and makes use of the quantum minimum searching subroutine and the quant_find_smallest_values subroutine among other quantum routines. The details are described in the next subsections.

3.1 Data storage in quantum random access memory

QRAM (Giovannetti et al. 2008) is used to store the input data of QOPTICS algorithm. As input data, the algorithm handles the distance matrix D containing the distances between objects in addition to the parameters *eps* and *MinPts*. To store the matrix D in QRAM, an address register and an output register are defined. As there are n^2 elements in D, the address register is composed of qubits and is a superposition of n^2 addresses as shown in Eq. 2.

$$\sum_{k=0}^{k=n^2-1} \frac{1}{n}|k>_a \tag{2}$$

The output register is also composed of qubits and is a superposition containing the distances correlated to the address register as shown in Eq. 3.

$$\sum_{k=0}^{k=n^2-1} \frac{1}{n}|k>_a \xrightarrow{QRAM} \sum_{k=0}^{k=n^2-1} \frac{1}{n}|k>_a |dist_{i,j}>_r \tag{3}$$

where $k = (i-1)*n+j-1$.

With QRAM, it takes $O(\log n^2)$ that is $O(\log n)$ time to access a distance as there are $\log n^2 = 2\log n$ number of qubits in the address register.

The output of QOPCTICS is an ordered file where the objects are ordered within the clusters they belong to. Each object contains its core-distance and its reachability distance. The file can be stored using a superposition of address and output register as shown in Eq. 4.

$$\begin{aligned} &\sum_{k=0}^{k=n-1} \frac{1}{\sqrt{n}}|k>_a \xrightarrow{QRAM} \\ &\sum_{k=0}^{k=n-1} \frac{1}{\sqrt{n}}|k>_a |core_k>_r |reachability_k>_r \end{aligned} \tag{4}$$

where core and reachability are two different superpositions of n items.

3.2 QOPTICS algorithm

The algorithms below highlight the theoretical constructs of the quantum version of QOPTICS using quantum subroutines such as *Quant-Min-Search* (Durr and Hoyer 1996) and *Quant_find_smallest_values* (Durr et al. 2006). Algorithm 1 is the quantum algorithm of OPTICS. It calls the quantum subroutines Algorithm 2 and Algorithm 3. Algorithm 2 produces $MinPts$ neighbors of an object p that are away from p by a distance equal to eps. Algorithm 3 calculates the core distance of the current object and the current reachability distance of the objects existing in *eps-Neighbors*. *Seeds* has no needs to be ordered as the quantum subroutine *Quant-Min-Search* can be invoked to retrieve the object with the minimum reachability distance. A quantum oracle O is created and used in Algorithm 3 to transform the index $|q>$ belonging to *eps-Neighbors* into a state $|q.reachability>$ defined as in Eq. 5.

Algorithm 1 QOPTICS Algorithm

input: D, eps, $MinPts$ /* D is the dataset and eps and $MinPts$ are parameters
output: Ordered list of Clusters of objects with their core and reachability distances

for each point p of D not processed **do**
  p.processed = *true*
  eps-Neighbors = **Quant-eps-Neighbors**(p, eps)
  p.reachability = *undefined*
  p.core = **Quant-Core-Distance**(p, $MinPts$, eps-Neighbors)
  OrderedList[p] = (p.core, p.reachability)
  Seeds = *empty*
  if p.core $\neq$ *undefined* **then**
    for each q **in** eps-Neighbors **do** Seeds = Seeds $\cup$ {q, q.reachability}
    end for
  end if
  while Seeds not empty **do**
    Quant-Min-Search(p, Seeds) /* p has the minimum reachability distance
    Seeds = Seeds - {p, p.reachability} /* retrieve p from Seeds
    p.processed = *true*
    eps-Neighbors = **Quant-eps-Neighbors**(p, eps)
    p.reachability = *undefined*
    p.core = **Quant-Core-Distance**(p, $MinPts$, eps-Neighbors)
    OrderedList[p] = (p.core, p.reachability)
    if p.core $\neq$ *undefined* **then**
      for each q **in** eps-Neighbors **do** Seeds = Seeds $\cup$ {q, q.reachability}
      end for
    end if
  end while
end for
Store in QRAM (OrderedList, p, p.core, p.reachability)

Algorithm 2 Quant-eps-Neighbors

input: D, p, eps
output: eps-neighbors of p, set of points q such that $dist_{p,q}$ <= eps

eps-neighbors = *empty*
built = *false*
D' = D
while not built **do**
  using **Quant-Min-Search**, find q of D' such that $dist_{p,q}$ is minimum
  if $dist_{p,q}$ <= eps **then**
    eps-neighbors = eps-Neighbors $\cup$ {q}
    retrieve q from D'
  else built = *true*
  end if
end while
return eps-Neighbors

Algorithm 3 Quant-Core-Distance

input: D, p, $MinPts$, eps-Neighbors
output: core-distance of p

using **Quant_find_smallest_values**, determine MinPts-Neighbors, the $MinPts$ nearest neighbors of p
p.core = max($dist_{p,q}$), q belonging to MinPts-Neighbors
using **Hadamard transform**, create the uniform superposition of all states $dist_{p,q}$ for q in eps-Neighbors
using the **oracle** O, determine q.reachability for all q in MinPts-Neighbors
return p.core

$$q.reachability = \begin{cases} p.core & \text{if } dist_{p,q} \leq p.core \\ dist_{p,q} & \text{otherwise} \end{cases} \quad (5)$$

Lemma 1 *The computational complexity of Quant-eps-Neighbors subroutine is* $O(logn\sqrt{n})$.

Proof *Quant-eps-neighbors* Algorithm calls *Quant-Min-Search* subroutine a number of times equal to the number of *eps-neighbors*. The worst-case access time to *eps-neighbors* is $O(logn)$ as the maximum size of *eps-neighbors* is n. The computational complexity of *Quant-min-search* is $O(\sqrt{n})$, then the computational complexity of *Quant-eps-Neighbors* is $O(logn\sqrt{n})$. □

Lemma 2 *The run time of Quant-Core-Distance subroutine is* $O(\sqrt{MinPts \times n})$.

Proof Algorithm 3 calls *Quant_find_smallest_values* in $O(\sqrt{MinPts \times n})$ according to Durr et al. (2006). The reachability distance is calculated for each object in MinPts-neighbors in $O(logn)$ time. The Hadamard transform is computed in $O(1)$ time, as it is a quantum logic gate. The application of a Hadamard gate to each qubit of an n-qubit register in parallel is equivalent to the Hadamard transform H_n. This operation is obtained in $log(n)$ steps. The computational complexity of *Quant-Core-Distance* Algorithm is then $O(\sqrt{MinPts \times n})$. □

Theorem 1 *The running time of QOPTICS is* $O(logn\sqrt{MinPts \times n})$.

Proof QOPTICS calculates for each point its *eps-neighborhood* with a complexity of $O(logn\sqrt{n})$. As the worst case of the *eps-neighborhood* considered in Lemma 1 was n, this complexity stands also for all the objects. The algorithm computes also for each object its core distance and its reachability distance with a complexity of $O(\sqrt{MinPts \times n})$. As the points can be accessed in $O(logn)$, its time complexity is then $O(logn\sqrt{MinPts \times n})$.

In order to highlight the gain in execution time brought by QOPTICS, let consider the case of our dataset, where n is the number of hospitals, which is equal to 279 and MinPts = 1 hospital. If the unit of time is 1 millisecond, then OPTICS runtime is 279^2= 77 841 milliseconds, which is equal to 77 seconds and QOPTICS runtime is $log279\sqrt{279}$ = 16 milliseconds or 0,016 seconds. The gain is even more significant for larger datasets. □

Theorem 2 *The spatial complexity of QOPTICS is* $O(log_2 n)$.

Proof QOPTICS uses a QRAM to store the distances from two distinct points in a register, which is a superposition of the distances. Its size is $O(log_2 n^2)$, as the number of distances that separate two points is $O(n^2)$. It is then $O(log_2 n)$ as $log_2 n^2 = 2log_2 n$. The reachability distance is calculated by an oracle and is stored in memory of size $log_2 n$, as the reachability distance is calculated for each point. Therefore, in total the spatial complexity of QOPTICS is $O(log_2 n)$. □

4 Deep self learning AOA and EHO

This section presents the proposed Deep Self-Learning approach on Artificial Orcas Algorithm (AOA) (Bendimerad and Drias 2020) and on Elephant Herding Optimization (EHO) (Wang et al. 2015). First, AOA is briefly described followed by its improved version using deep self-learning. The deep self-learning EHO is based on the same principle.

4.1 Artificial Orca algorithm (AOA)

Swarm Intelligence Algorithms have received a lot of attention these last decades. In Bendimerad and Drias (2020), the authors proposed one of these algorithms mimicking orcas in their living environment. Recently, in order to add the cultural dimension of orcas, AOA was hybridized with the cultural algorithm (CA) to develop an algorithm called OCA (Drias et al. 2021). The social organization of orcas includes several clans containing in their turn pods of individuals. Orcas practice echolocation to detect preys and perform various types of hunting strategies to reach their prey. All these phenomena are simulated in an algorithm called Orcas Artificial algorithm (AOA). The artificial orcas are directed by a matriarch which is considered as the fittest individual in the pod. In addition to this, each hierarchical structural level is distinguished by a degree of closeness. The pods are closer to themselves than to the clans. AOA is featured by making an excellent balance between two very important phases of evolutionary algorithms; the search intensification and the search diversification. The Original AOA is outlined in Algorithm 4.

Algorithm 4 Artificial Orcas Algorithm AOA

input: D, empirical parameters
output: an optimal solution or a solution of high quality
1: Using the empirical parameters of the social structure that is the number of individuals per pod, the number of pods per clan, the number of clans in a population and the distance separating the different hierarchical levels, initialize a population.
2: Evaluate the population by calculating the fitness of all its individuals.
3: Sort the individuals of the pods w.r.t their fitness value and determine the matriarch of each pod.
4: Update the individuals fitness using the intensification search, composed of the echolocation search represented in Equations 6, 7 and 8 and the hunting strategy update using Equations 9, 10 and 11.
5: Update the worst individual of each pod using Equation 12.
6: Evaluate the new individuals by calculating their fitness value.
7: **if** stopping criteria is reached **then**
8: **return** the best individual of the population else go to 3.
9: **end if**

$$f_{group} = f_{min} + (f_{max} - f_{min}) \tag{6}$$

$$v_{p_i}^t = v_{p_i}^{t-1} + f_i \times D_p + f_c \times D_c + f_{pop} \times D_{pop} \tag{7}$$

$$x_{p_i}^t = x_{p_i}^{t-1} + v_{p_i}^t \tag{8}$$

$$x_{temp,p_i}^t = A \times \sin\left(\frac{2 \times \Pi}{L} \times x_{p_i}^{t-1}\right) \times \cos\left(\frac{2 \times \Pi}{T} \times t_x\right) \tag{9}$$

$$x_{m,p}^t = \frac{\sum_{j=1}^{j=n} x_{temp,p_j}}{n} \tag{10}$$

$$x_{p_i}^t = x_p^* - \beta \times x_{m,p}^t \tag{11}$$

$$x_{new,p_i}^t = \frac{\gamma \times x_{pop_{rand1}} + \omega \times x_{c_{rand2}}^t}{2} \tag{12}$$

t indicates the current iteration. $x_{p_i}^t$ is the individual at position i in the pod p. $v_{p_i}^t$ is the velocity of the individual at position i in the pod p. f_{min} and f_{max} are respectively the minimum and the maximum frequencies and are used to generate a random frequency f_{group} in this range. Note that $group$ corresponds either to p, c or pop in order to determine the frequency for the pod, the clan and the population to which the individual belongs. α, γ and ω are random numbers in interval [0,1]. The population levels are respectively distant from each other by the distances D_p, D_c and D_{pop} defined as follows.

- $D_p = |x_{p_i}^{t-1} - x_p^*|$ where x_p^* is the matriarch of the pod to which $x_{p_i}^{t-1}$ belongs.
- $D_c = |x_{p_i}^{t-1} - x_c^*|$ where x_c^* is the matriarch of the clan to which $x_{p_i}^{t-1}$ belongs.
- $D_{pop} = |x_{p_i}^{t-1} - x_{pop}^*|$ where x_{pop}^* is the matriarch of the population.

For the hunting strategies, A is a parameter depending on the problem modeling, L is a parameter representing the wave length and T is an empirical parameter that represents the wave period during the chasing activity. x^t_{new,p_i} is the new solution of the individual i in the pod p. $x^t_{pop,rand1}$ is a random individual in the population at position $rand1$ and $x^t_{c,rand2}$ is a random individual in the clan c of the current pod at position $rand2$.

4.1.1 Analysis of AOA complexity

The number of operations of AOA denoted TC_{AOA} and presented in Eq. 13 depends on the maximum number of iterations $MaxIter$, the population size n and the number of pods $\#pods$. Note that the sorting method used in the algorithm is *heapsort* and its computational complexity is $O(nlogn)$.

$$TC_{AOA} = \sum_{i=1}^{MaxIter} (nlogn + n + \#pods + n) \tag{13}$$

$nlogn$ operations are needed for Instruction 3, n operations for Instruction 4 and Instruction 6 and $\#pods$ operations for Instruction 5. As the size of the population $n = \#clans \times \#pods \times \#orcas$, $n > \#pods$, we conclude that:

$TC_{AOA} < \sum_{i=1}^{MaxIter} (nlogn+3n)$ and therefore the computational complexity of AOA is $O(MaxIter \times nlogn)$

4.2 Elephant herding optimization (EHO)

Elephant Herding Optimization (EHO) is a Swarm Intelligence-based method inspired by the herding behavior of elephants and proposed to solve optimization problems. Since it was firstly proposed in 2015 by Wang et al. (2015), it has received significant attention from scholars in the world (Li et al. 2020). Several EHO improvements and hybridization have been recently published (Tuba et al. 2018; Li et al. 2019; Moayedi et al. 2020; Houacine and Drias 2021).

In nature, elephants have a social structure composed of clans, where the elephants of each clan are under the leadership of a matriarch. Females live in family clans, while male elephants leave their group once they grow up. These two behaviors are formalized into EHO through two main operators: *Clan updating operator* and *Separating operator*.

Also, EHO is characterized by a maximum number of generations (iterations), the influence rate of the matriarch on elephants (α), and the influence rate of the clan's gravity center on the matriarch (β). EHO algorithm is summarized in Algorithm 5.

Algorithm 5 Elephant Herding Optimization EHO

input: D, empirical parameters
output: an optimal solution or a solution of high quality
1: Using the empirical parameters of the social structure that is the number of elephants per clan and the number of clans in a population, initialize a population.
2: Evaluate the population by calculating the fitness of all its individuals (elephants).
3: Sort the elephants of the clans w.r.t their fitness value and determine the matriarch and the male elephant of each clan.
4: Update the best elephant of each clan using Equation 14 and 15.
5: Update the worst elephant of each clan using Equation 16.
6: Update the rest of elephants of each clan using Equation 17.
7: Evaluate the new elephants by calculating their fitness value.
8: **if** stopping criteria is reached **then**
9: **return** the best individual of the population else go to 3.
10: **end if**

$$x^{t+1}_{i,j} = \beta \times x^{t+1}_{center,j} \tag{14}$$

$$X_{center,i} = \frac{1}{\#E} * \sum X_{i,j} \tag{15}$$

$$x^{t+1}_{worst,j} = x_{min} + (x_{max} - x_{min}) \times rand \tag{16}$$

$$x^{t+1}_{i,j} = x^t_{i,j} + \alpha \times (x^{t+1}_{best,j} - x^t_{i,j}) \times r \tag{17}$$

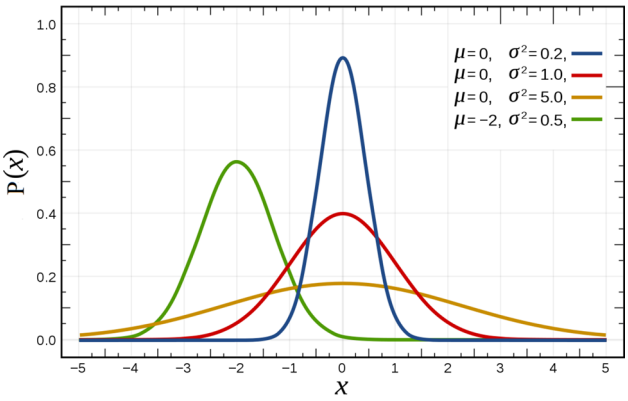
$\#C$ and $\#E$ are the number of clans and the number of elephants in each clan, respectively. $x^t_{i,j}$, $x^{t+1}_{i,j}$, respectively represent the actual and newly updated i-th elephant's solution of the j-th clan. $x^t_{best,j}$, $x^t_{worst,j}$ are the best and the worst elephant's solution of the j-th clan. x_{max}, x_{min} are the upper and lower bound of a solution, $x_{center,j}$ gives the gravity center of clan j, $rand$, r are random numbers $\in$ [0,1], and α, $\beta \in$ [0,1].

4.2.1 Analysis of EHO algorithm complexity

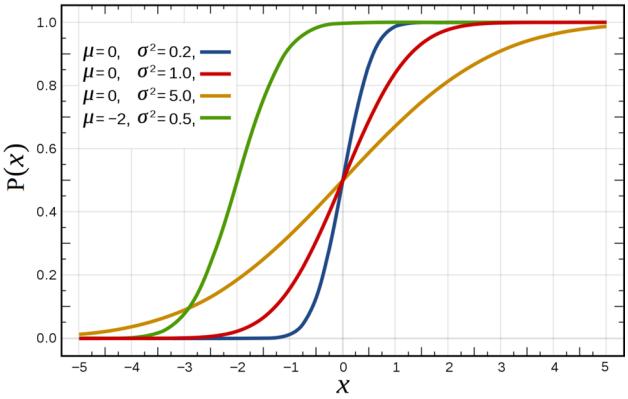
The number of operations of EHO algorithm at worst is given in Eq. 18 and is based on the maximum number of iteration $MaxIter$, the population size n which is equal to the product of the number of clans $\#C$ by the number of elephants $\#E$.

$$TC_{EHO} = \sum_{i=1}^{MaxIter} \#C \times \#E \times \log(\#E) + 2n + const \tag{18}$$

$\#C \times \#E \times \log(\#E)$ operations are needed for Instruction 3, n operations for Instruction 6 and Instruction 7 and a constant $const$ for Instruction 4 and Instruction 5. We deduce that:
$TC_{EHO} = MaxIter \times (nlog(\#E) + 2n + const)$. The computational complexity of EHO algorithm is then $O(MaxIter \times nlog(\#E))$.



(a) Gaussian Distribution



(b) Cumulative Gaussian distribution

Fig. 3 Gaussian distribution in interval $x \in [-5, 5]$

4.3 Mutation based on Gaussian and Cauchy distributions

Evolutionary algorithms are known to have limited potential to solve very complex real-world problems (Kechid and Drias 2020). One possible way to strengthen their performance and to allow them to escape from local optima is to introduce within the algorithm self-learning strategies (Wang et al. 2020) that have proved recently their usefulness. Gaussian and Cauchy distributions are two operators used for that purpose.

4.3.1 Mutation based on Gaussian operator

The theoretical basis of Gaussian mutation (Jakubik et al. 2021) is based on the Normal density probability function defined in Eq. 19 and depicted with its cumulative distribution function in Fig. 3.

$$P(x) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) \tag{19}$$

μ represents the mean, σ the standard deviation and erf the error function, which is defined in Eq. 20.

$$erf(x) = \frac{2}{\sqrt{\Pi}}\int_0^x e^{-t^2}.dt \tag{20}$$

4.3.2 Mutation based on Cauchy distribution function

Similarly, the Cauchy mutation operator (Paiva et al. 2017) is based on the Cauchy distribution function presented in Eq. 21 and shown with its cumulative function in Fig. 4.

$$P(x) = \frac{1}{2} + \frac{1}{\Pi}\arctan\left(\frac{x-x_0}{\gamma}\right) \tag{21}$$

DSLAOA and DSLEHO

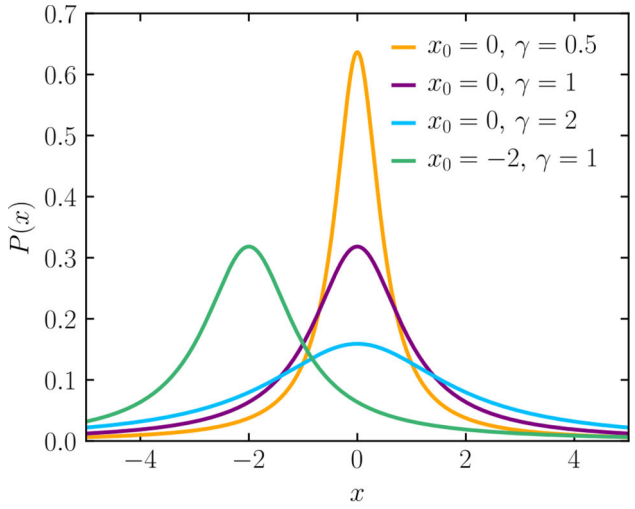
The designed deep self-learning on AOA and EHO are called respectively Deep Self-Learning AOA (DSLAOA) and Deep Self-Learning EHO (DSLEHO). In what follows, the approach is detailed for AOA. The mutation operators are integrated in AOA after the fourth step of the algorithm between the intensification and the diversification phases. The matriarch of each pod is deeply improved using Eq. 22 expressed in terms of two random individuals of this pod. In addition, a new parameter called *Depth* is introduced to define the depth of learning. For EHO, The matriarch of each clan is deeply improved using the same equation.

$$x^t_{new,p_{matriarch}} = MO \times (x^t_{p,rand1} - x^t_{p,rand2}) + x^t_{p_{matriarch}} \tag{22}$$

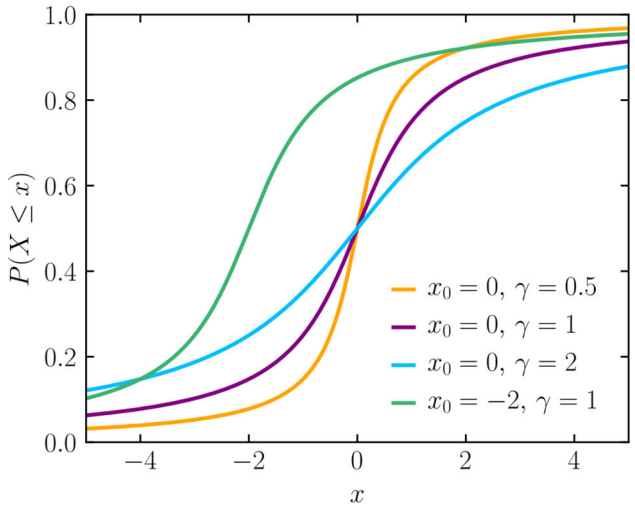
MO corresponds either to Gaussian Mutation Operator or Cauchy Mutation Operator. Algorithm 6 outlines the Deep Self-Learning algorithm.

Algorithm 6 Deep Self-Learning for AOA and EHO

input: *Depth* parameter, Solution S, Solution Dimension k
output: Solution S after improvement
1: **for** i = 1 to *Depth* **do**
2: **for** j = 1 to k **do**
3: Calculate $S_{new}[j]$ by applying Equation 22 for current k.
4: **end for**
5: **if** S_{new} is better than S **then**
6: $S = S_{new}$
7: **end if**
8: **end for**
9: **Return** S



(a) Cauchy Distribution



(b) Cumulative Cauchy distribution

Fig. 4 Cauchy distribution in interval $x \in [-5, 5]$

4.3.3 Analysis of DSLAOA algorithm complexity

The computational complexity of Algorithm 6 depends greatly on the depth parameter $Depth$. This parameter involves adding a loop to AOA to repeat applying Eq. 22 to the matriarch of each pod of the population. The computational complexity of DSLAOA is then obtained by adding to the complexity of AOA the operations of the introduced loop, that is $\#Clans \times \#pods \times Depth$. It is then $O(MaxIter(nlogn + \#Clans \times \#pods \times Depth)$, Which is at worst $O(MaxIter \times nlogn \times Depth)$.

The $Depth$ parameter, as its name suggests, is supposed to be a large number that allows the improvement of the best solution due to several executions of the mutation operators (GAUSS or CAUCHY). Accordingly, the complexity of DSLAOA is more likely to be greater that that of AOA.

4.3.4 Analysis of DSLEHO algorithm complexity

With similar reasoning as that done for the computational complexity calculation of DSLAOA, we obtain that the computational complexity of DSLEHO algorithm is at worst $O(MaxIter \times nlog(\#E) \times depth)$.

5 Adaptation of DSLAOA and DSLEHO for Covid-19 EMS transportation

In this section, the Covid-19 EMS transportation problem is described and formulated mathematically. Thereafter, the adaptation of DSLAOA and DSLEHO for the issue is presented.

5.1 Related work

Ambulance fleet management is an essential tool to avoid severe problems in emergency medical services (EMS). Early studies have been undertaken for the issue of the emergency vehicle dispatching problem with different static approaches based among others on the minimization of the total travel time in the system, the priority of the call and the first come first served strategy (Bandara et al. 2014; Lee 2017). On the other side, the covering demand problem has also been investigated using static models. These approaches are well reported in a recent survey (Belanger et al. 2019), which focuses on operations research (OR) approaches applied to EMS. Researchers are still interested in this issue as it can be applied to fire departments, police stations and human rescue in natural disasters such as earthquakes. Very recent works (Belanger et al. 2020; Usanov et al. 2019) and (Carvalho et al. 2020) proposed new methods based on OR strategies such as Markov decision process and heuristic information. Interesting case studies were realized for Amsterdam and Lisbon.

As far as we know, few studies have been dedicated to integrate the ambulance dispatching and the demand covering problems jointly in the same model. Except the work held in Ibri et al. (2012), the majority of the efforts have adopted OR approaches with simple heuristics. None of the previous works have used sophisticated bio-inspired approaches such as DSLAOA or DSLEHO. Also, none of them have explored the decomposition of the territory into sub-territories with a data science technique such as the OPTICS clustering method.

5.2 EMS problem formulation

The main target of the problem is to assign an ambulance to each emergency call so that there is no unanswered call in due time. To tackle the issue, the problem is modeled as follows.

- The day is a sequence of several periods of times p, representing respectively the calls arrival times.
- A list C of the calls that arrive at each period of time. The calls are specified by their positions and their priority.
- A list S of stations defined by hospital positions.
- Each station contains a list A of ambulances, among them only Z ambulances are available.
- A list E of edges representing the distance between a call $c \in C$ and a station $s \in S$.
- The travel time t_{a_s} that keeps an ambulance a_s belonging to Station s busy is expressed by Eq. 23.

$$t_{a_s} = 2 \times \left(UnitPerHour \times \frac{Distance(a_s, c)}{v_s} + p \right) \quad (23)$$

c is the call to which a_s is responding. v_s represents the emergency vehicle speed (in km/h) and $UnitPerHour$ is the number of time units in 1 h. p represents the time it takes to deal with the emergency and distance(c, s) is the spherical distance between two geographical points calculated by Eq. 24.

$$distance(c, s) = 2 \times \arcsin \sqrt{\sin\left(\frac{\delta_c - \delta_s}{2}\right)^2 + \cos\delta_c \times \cos\delta_s \times \sin\left(\frac{\gamma}{2}\right)^2} \times R \quad (24)$$

δ and γ represent, respectively, the radians of the latitude and longitude of each point and R the earth radius. Based on these assumptions, the problem can be defined by the multi-objective function of Eq. 25.

$$F = Obj_1 + Obj_2 + Obj_3 \quad (25)$$

Obj_1 is to minimize the predicted travel time, which can be expressed as: $\sum_{j \in S} \sum_{i \in C} t_{i,j} \times x^t_{i,j}$

Obj_2 is to minimize waiting calls, which can be expressed as:
$\sum_{i \in C} q^t_i$

Obj_3 is to minimize unserved priorities, which can be formulated as:
$\sum_{i \in C} Priority_i \times q^t_i$
where:

$$x^t_{i,j} = \begin{cases} 1 & \text{if an ambulance from Station } j \text{ is assigned to Call } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$q^t_i = \begin{cases} 1 & \text{if Call } i \text{ is waiting for an assignment of an ambulance} \\ 0 & \text{otherwise} \end{cases}$$

These objective functions are subject to the following constraints:

1. Assignment uniqueness: $\sum_{i \in C} x^t_{i,a} + \sum_{j \in C} x^t_{j,a} = 1$.
2. Reachability constraint: $E(i, a) \neq \phi$ where i represents the position of the incoming call and a an ambulance assigned to i.

5.3 Solution description

A solution S is a sequence of sub-solutions, where each sub-solution x_i represents a pair of points (c_i, a_j), c_i belongs to C and a_j to an ambulance of Station j. A sub-solution (Call, Ambulance) is characterized by its geographical position (Longitude, Latitude). AOA, DSLAOA, EHO and DSLEHO deal with the update of the solution using arithmetic operators. Adapting these operators to the current solution consists in updating the latitude and longitude of the call and the ambulance of each sub-solution.

5.4 DSLAOA and DSLEHO for Covid-19 EMS transportation

The Covid-19 EMS problem we are dealing with is dynamic, that is, the emergency calls arrive in real-time at different period of time. For this purpose, the system updates the data (Covid-19 incoming calls and occupation time of ambulances) at each period of time. When calls arrive and ambulances are released, the developed algorithms calculate the best solution consisting in an assignment of an ambulance to a call respecting the multi-objective fitness function defined in Eq. 25. Algorithm 7 highlights the dynamic aspect of the approach that addresses DADECC-COVID19 problem.

Algorithm 7 DSLAOA and DSLEHO for DADECC-COVID19

input: Empirical Parameters of DSLAOA/DSLEHO, Dataset and List of different Unit-of-time
output: Solution S
1: **for** each Unit-of-Time t **do**
2: Update Dataset
3: incoming-calls = unsatified-calls[t-1]+Dataset[t]
4: update occupation time of ambulances
5: update the list of available ambulances of each hospital
6: Launch DSLAOA/DSLEHO
7: Best Solution= DSLAOA/DSLEHO (incoming-calls, available ambulances, empirical parameters)
8: **end for**

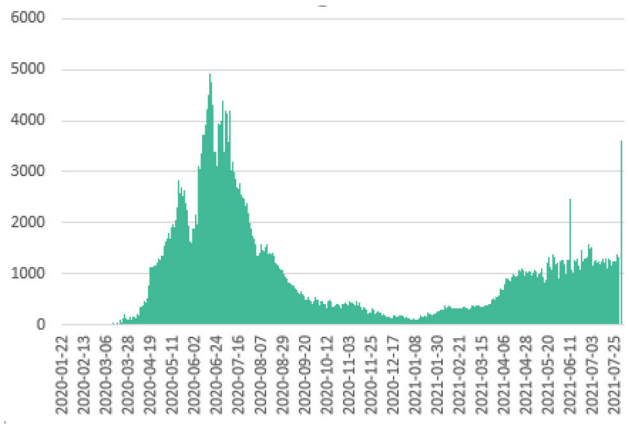
6 Experiments

QOPTICS is designed to help solving the NP-hard DADECC-COVID19 problem. An application has been developed to simulate the case of EMS transportation of KSA in the context of Covid-19 crisis. QOPTICS is used in a first phase of the program to locate the zones that condense an important number of hospitals. Also, the emergency calls have been clustered in order to treat the issue in one local region and thus to reduce this way the complexity of the problem solving. Once, these regions are identified, DADECC-COVID19 has been addressed using the intelligent methods DLSAOA and DSLEHO presented previously.

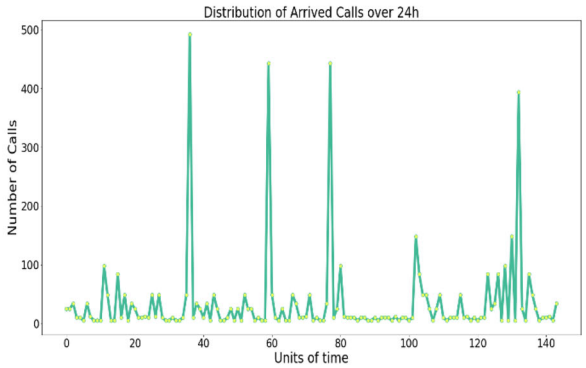
6.1 Datasets construction

The objects to be clustered are the KSA hospitals and the resulted outcome is the spatial clusters representing dense regions of hospitals. The dataset is built using the data information available on the Open Data portal of Saudi Arabia (2021). The data have undergone a preprocessing step where redundant data, maternity centers and mental patient centers were removed and GPS position of each hospital according to Google Maps was added. This way, an instance of the dataset is composed of a hospital and its geographic coordinates. As we are dealing with geolocation data, Eq. 24 was used to calculate the distance between two hospitals. The dataset containing the distances was generated to be used as input for QOPTICS.

The second dataset provides emergency calls we generated on the basis of the highest number of infected cases per day in Rami (2021). Figure 5a presents the evolution of the cases of Covid-19 pandemic noted until August 3, 2021. The peak encountered happened on June 17, 2020 with 4919 new cases. The emergency calls were built with these cases while respecting their distribution in their region as depicted in Fig. 5b. These calls are represented by their region, their geographical position and a priority of call as (Region, (Longitude, Latitude), Priority). The datasets we built can be found in Datasets for DADCC (2021).



(a) Distribution of Covid-19 cases in Saudi Arabia



(b) Distribution of the number of emergency calls

Fig. 5 Dataset construction

6.2 Experimental results

The performed experiments consist of two parts. First, we present the clustering results to validate QOPTICS and then the outcomes of the execution of DSLAOA and DSLEHO applied to Covid-19 emergency transportation.

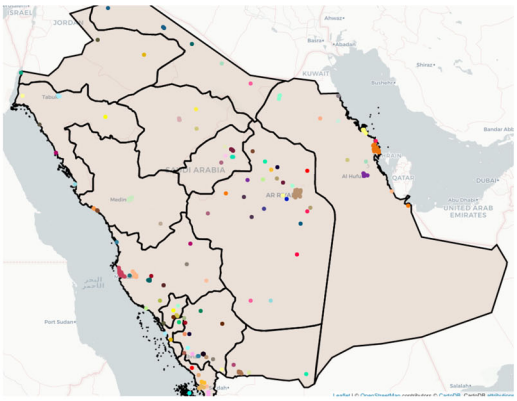
6.2.1 Clustering results

As all hospitals should be exploited for hosting the calling patients, outliers are unauthorized. Therefore, $MinPts$ is set to 1. To fix *eps* the distance that determines the *eps-neighbors*, various tests were performed on values of $MinPts$ (the number of hospitals in the *eps-neighborhood*) ranging from 5 to 40 km by step equal to 5. The experimental outcomes are shown in Table 1.

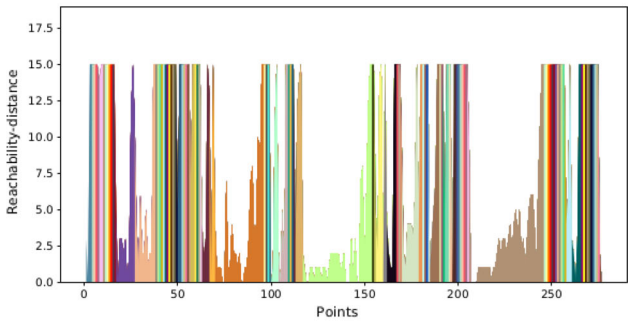
Table 1 Parameters ' setting for clustering hospitals

MinPts	5	10	15	20	25	30	35	40
eps	144	119	108	98	91	81	75	72

The best setting is the one that minimizes both the number of clusters and *eps* in order to make the calls positions close to the hospitals. According to the results presented in Table 1, the most appropriate setting would be $eps = 15$ km, which allows the formation of 108 clusters. The dispersion of the 108 clusters on the KSA map is shown in Fig. 6a. Each cluster is represented by a distinct color. The reachability plot resulting from clustering with the best parameterization found is depicted in Fig. 6b, which shows the structure of the data in clusters with different densities. The correspondence between the representation of the 50th cluster (which includes 43 hospitals) on the map and on the reachability plot is illustrated in Fig. 7.



(a) Geographic map of hospitals clusters with $MinPts$=1 and eps =15km



(b) Reachability plot for hospitals with eps =15km

Fig. 6 Hospitals clustering with $MinPts$ = 1 hospital and eps = 15 km

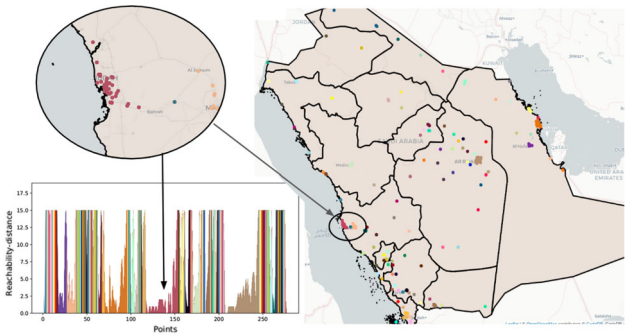
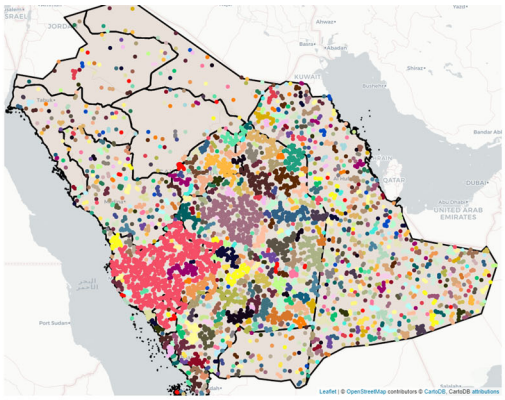


Fig. 7 A mapping of a hospitals cluster between its geographic position and the reachability plot

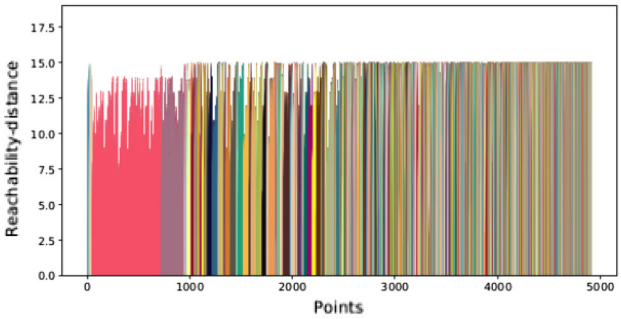
The second set of experiments concern the clustering of the Covid-19 emergency calls. As for hospitals clustering, we varied eps from 5 to 40 by step = 5 and tested the algorithm to tune this parameter. The results of the experiments are reported in Table 2. According to these outcomes, the clustering of the 4919 emergency calls that seems the most suitable is obtained with $Minpts$ = 1 and eps = 15, which correspond to 1237 clusters. Figure 8a and b shows respectively the clusters in the geographic map and the reachability plot. The mapping between a cluster and the reachability plot is depicted in Fig. 9.

Table 2 Parameters ' setting for clustering emergency calls

MinPts	5	10	15	20	25	30	35	40
eps	4210	2619	1237	518	248	124	80	63



(a) Geographic map of calls clusters with $MinPts$ = 1 and eps =15km



(b) Reachability plot for calls with eps =15km

Fig. 8 Calls clustering with $MinPts$ = 1 hospital and eps = 15 km

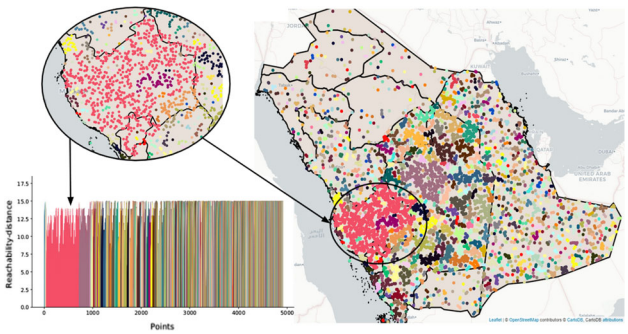


Fig. 9 A mapping of a calls cluster between its geographic position and the reachability plot

Table 3 Empirical parameters of deep self-learning AOA

Population size	60
Number of clans	5
Number of pods per clan	2
Number of orcas per pod	6
Wave length L	10
Wave period T	1000
$f_{\min}$	0
$f_{\max}$	1
The maximum number of iterations	20
The learning depth	100

Table 4 Empirical parameters of deep self-learning EHO

Population size	150
Number of clans	15
Number of individuals per clan	10
α	0.6
β	0.8
The maximum number of iterations	50
The learning depth	100

6.2.2 Results of deep self-learning AOA and EHO

The empirical parameters of AOA and EHO and the depth parameter of the Deep Self-learning have been first tuned. The values of these parameters for AOA are shown in Table 3 while those of EHO are presented in Table 4.

Recall that the Deep Self-Learning framework is based on two mutation operators: Cauchy and Gaussian. The algorithms that exploit the Cauchy operator are called DSLAOA-C and DSLEHO-C while the algorithms that are based on the Gaussian operator are called DSLAOA-G and DSLEHO-G. For better describing the dispatching mechanism, we focused on the region with the largest cluster of calls. the following tasks illustrate the whole process.

Step 1. The ambulance base formation.
Each cluster of hospitals is represented by a centroid to form a base. The resources of hospitals in the same cluster are grouped together and shared between them. Thus, the center of gravity of each hospital cluster is used to represent an ambulance base. Figure 10 takes as example the blue cluster of 37 hospitals whose center of gravity is estimated at (Latitude: 20.44837719, Longitude: 40.879243798).

Step 2. Emergency calls assignment to ambulances bases.
In order to decentralize and parallelize the dispatching system to different KSA regions, it was necessary to distribute a certain number of bases per emergency call cluster. For this purpose, the databases and call clusters have been grouped together. As the dispatching at the level of the different clusters is independent and carried out in parallel, we have chosen to treat the largest cluster shown in pink in Fig. 11. Figure 12 shows the result of regrouping the bases and emergency calls of the prominent cluster, this corresponds to 675 calls and 15 bases.

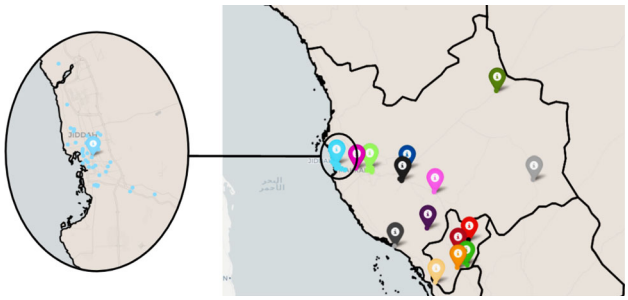


Fig. 10 Representation of each cluster of hospitals by an ambulance base

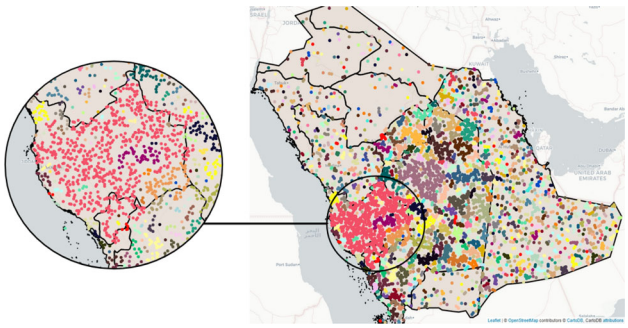


Fig. 11 Zoom in on the largest cluster of emergency calls

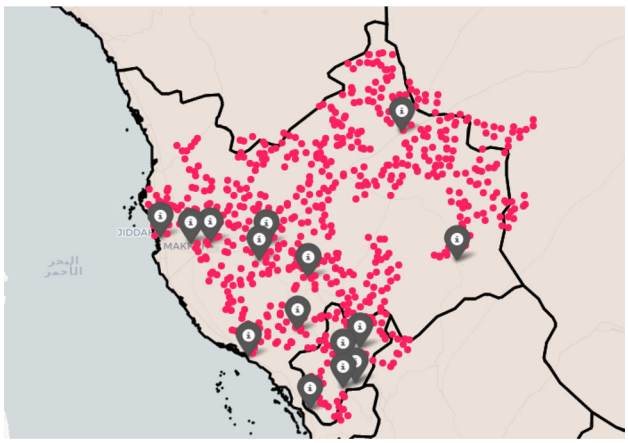


Fig. 12 Positions of the ambulance bases and the emergency calls of the prominent cluster

Step 3. DSLAOA/DSLEHO experimental results on the prominent cluster.
This step provides the results of the execution of DSLAOA/DSLEHO on the largest group (Calls, Ambulances). First the experiments undertaken on the parameter *depth* are exhibited followed by the outcomes yielded by these algorithms.

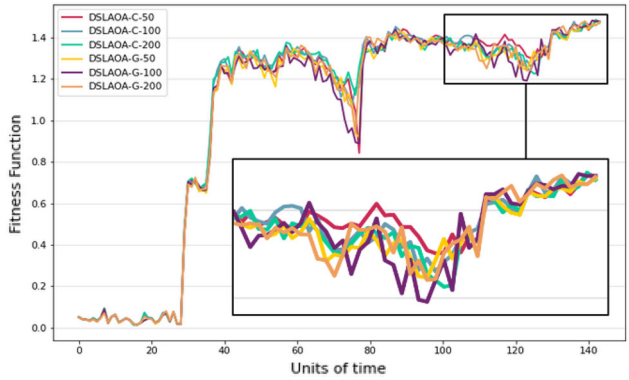


Fig. 13 Evolution of DSLAOA's overall fitness with different *Depth* values

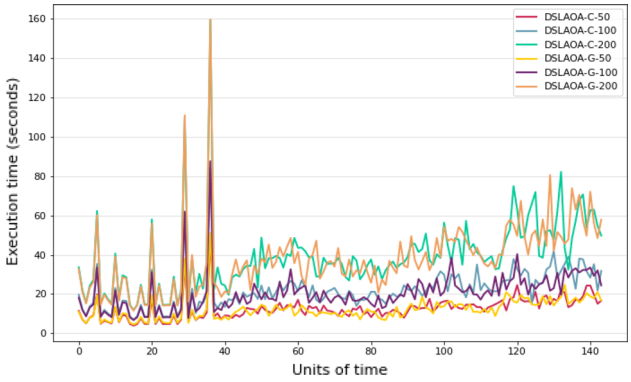


Fig. 14 Evolution of DSLAOA's execution time with different *Depth* values

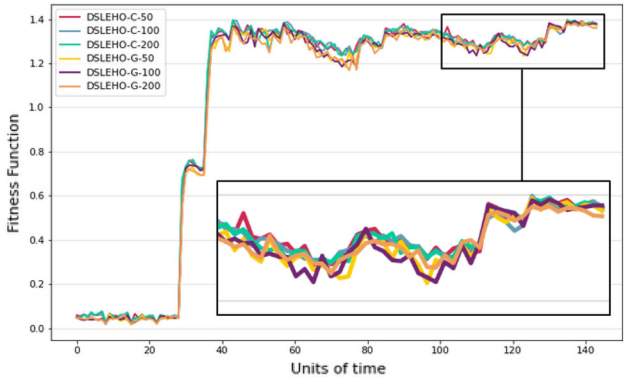


Fig. 15 Evolution of DSLEHO's overall fitness with different *Depth* values

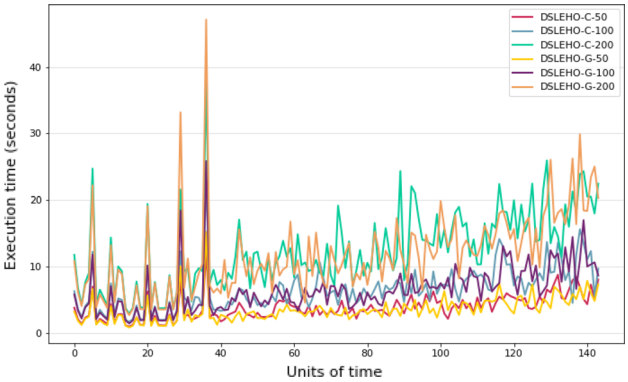


Fig. 16 Evolution of DSLEHO's execution time with different *Depth* values

A comparative study with a state-of-the-art method is performed at last.

Step 3.1. Impact of the parameter $Depth$.
In this part, we carried out experiments on the impact of the *Depth* parameter. We fixed the number of ambulances to 5 per base, and varied the *Depth* value in [50, 100, 200].

The influence of the *Depth* parameter on DSLAOA-C and DSLAOA-G in terms of performance and execution time are illustrated in Figs. 13 and 14, respectively. Figure 14 shows that the greater value of *Depth* the higher execution time for all the DSLAOA algorithms. However, Fig. 13 reveals that $Depth = 100$ performs the best results in almost all the time, with very few instances where its performance is equal to the one with *Depth* equal to 200. Consequently, we found out that the optimal *Depth* value is 100 in both terms performance and time execution.

Figures 15 and 16 illustrate the impact of *Depth* variation on DSLEHO-C and DSLEHO-G in terms of performance and execution time, respectively. Figure 16 shows that as for the DSLAOA results, the execution time increases with the growth of *Depth* value. While from Fig. 15, it appears that $Depth = 100$ and $Depth = 200$ perform the best results, with a negligible better fitness with $depth = 200$ until the 100th time unit. Then, from time unit 100 to 144, DSLEHO-C with $Depth = 100$ and 200 fluctuate in the same ranges and are equivalent. Thus, it comes out that a *Depth* value set to 200 does not improve sufficiently EHO performance to cover its growing time consumption. Therefore, setting *Depth* at 100 appears to be the best performance versus time ratio.
According to these experiments concerning both DSLAOA and DSLEHO, we set the depth to the best found value (100) for the remainder of the experiments.

Step 3.2. Comparing DSLAOA/DSLEHO with their origin models.
Original AOA and EHO as well as their Deep Self-Learning versions (DSLAOA-C, DSLAOA-G, DSLEHO-C, and DSLEHO-G) are compared on the basis of their execution time for the dispatching problem with 5 and 10 ambulances per bases.

Figures 17, 18, 19, and 20 show the execution time of AOA and EHO and their respective enhanced versions of deep self learning with both Cauchy and Gaussian mutation operators when dealing with 5 and 10 ambulances per base. It is clear that the deep approaches take much longer time than the original version of both AOA and EHO algorithms. These results reflect the accuracy of the theoretical study of

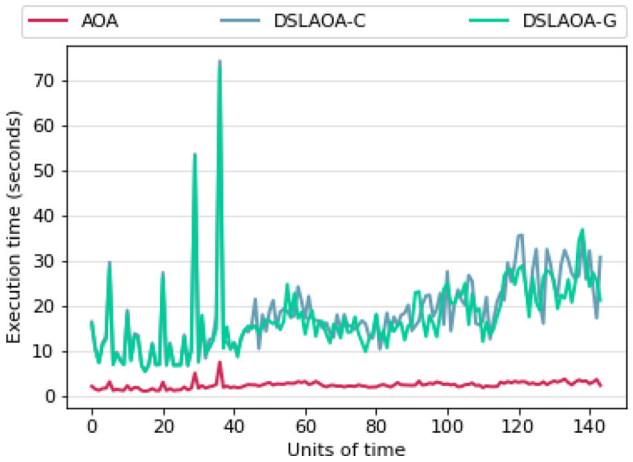


Fig. 17 Execution time of AOA and its improvement over 144 time units with 5 ambulances

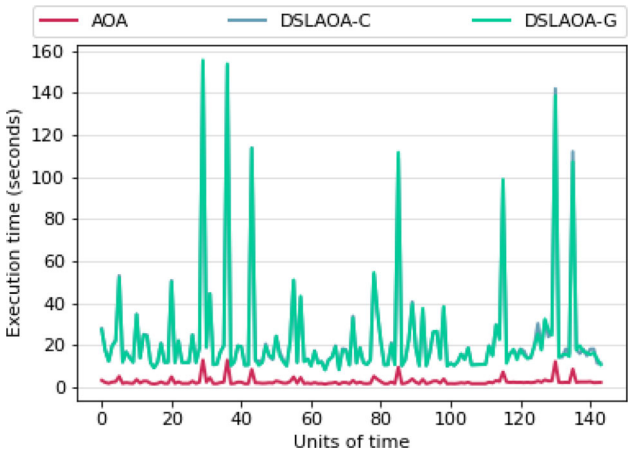


Fig. 18 Execution time of AOA and its improvement over 144 time units with 10 ambulances

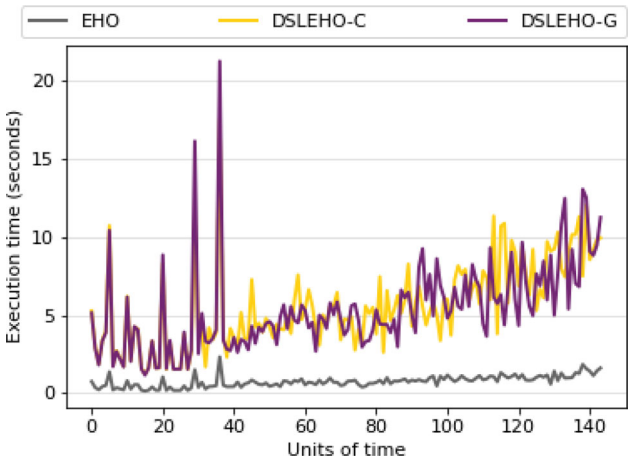


Fig. 19 Execution time of EHO and its improvement over 144 time units with 5 ambulances

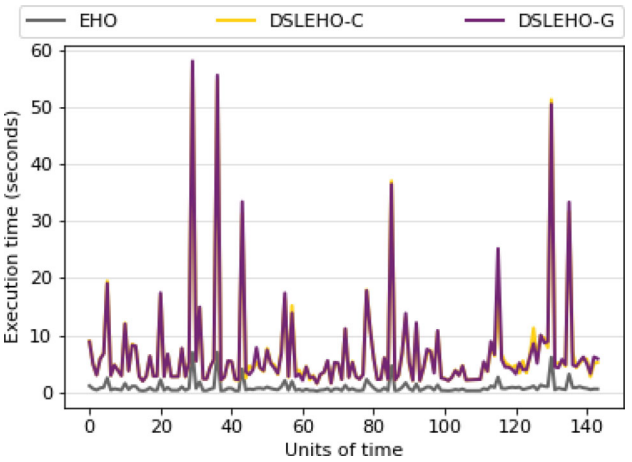


Fig. 20 Execution time of EHO and its improvement over 144 time units with 10 ambulances

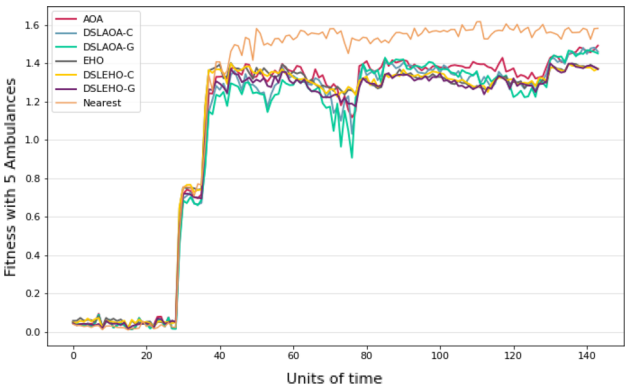


Fig. 21 Evolution of overall fitness over 144 time units (24h) with 5 ambulances per base

computational complexity. It shows that the proposed deep self-learning approaches have a disadvantage compared to the basic methods because it performs with a higher computational time (a quantity value of $\frac{Depth}{\#orcas}$ for AOA and $\frac{Depth}{\#E}$ for EHO). However, as long as this time problem remains polynomial, it can be easily circumvented either by switching to a GPU-based parallel implementation or by proposing a quantum approach.

Step 3.3. Comparing the six algorithms with a state-of-the-art method.

The performances of the six algorithms (AOA, DSLAOA-C, DSLAOA-G, EHO, DSLEHO-C, DSLEHO-G) were compared with the classical algorithm of the state-of-the-art (Closest-first policy) (Bandara et al. 2014). The executions were repeated 3 times on 5 different datasets. The overall fitness is calculated from the sum of the 3 sub-fitness Travel time, Unsatisfied calls and Priority of unsatisfied calls previously normalized between [0, 1] in order to balance the weight of each sub-fitness. The results of the minimization of the overall fitness spread over the 144 time units are pre-

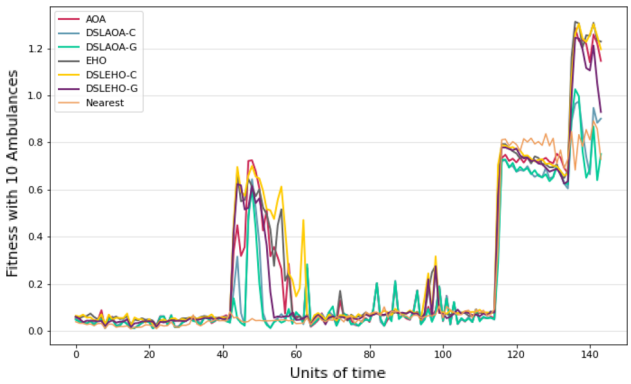


Fig. 22 Evolution of overall fitness over 144 time units (24h) with 10 ambulances per base

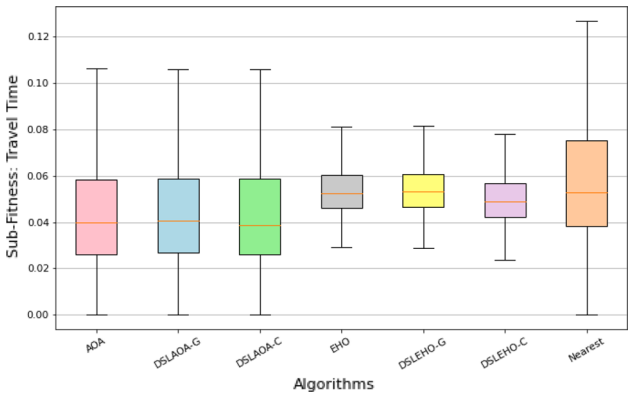


Fig. 23 Performances of the different algorithms for "Travel time" base

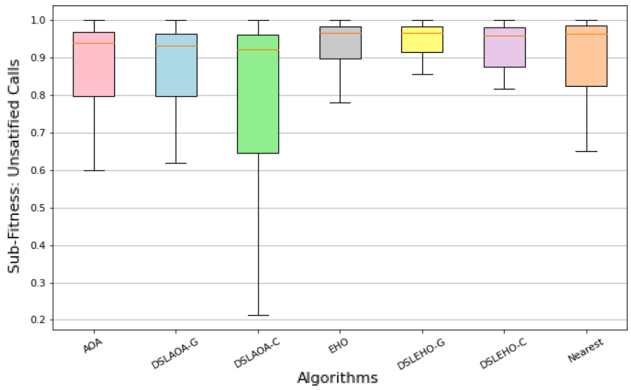


Fig. 24 Performances of the different algorithms for "Unsatisfied calls" with 5 ambulances per base

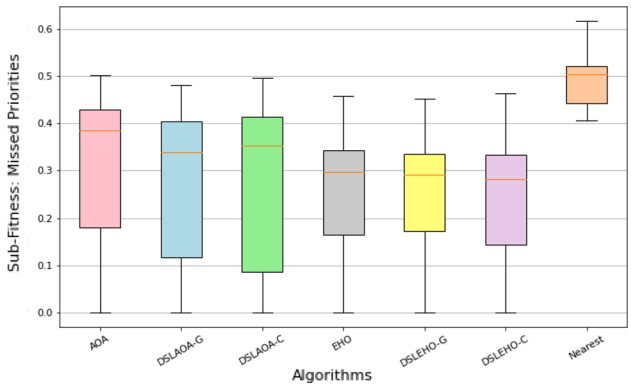


Fig. 25 Performance of the different algorithms for "missed priorities" with 5 ambulances per base

sented in Figs. 22 and 21. It is clear that in the long term when resources are limited (5 Ambulances per Base), the "Closest-first" (Nearest) algorithm records the highest fitness function scores, knowing that the latter should be minimized. When a large number of Covid-19 calls are received at the center level, the proposed Deep Self-Learning approaches perform better. These DSL-based approaches compete with each other, such that we remark very good results by DSLAOA (Cauchy and Gauss) up to time unit 79, where DSLEHO (Cauchy and Gauss) regain the upper hand.

Figure 21 shows over time units, that the Deep Self-Learning approaches with the two mutation operators Cauchy and Gauss improve the results obtained by AOA and EHO. Also, we note peaks at certain units of time: between the 40th and 60th and from the 118th unit, which corresponds to the period of high demand (urgent calls). Looking closely at the performance of the different algorithms, we find that the two variants of Deep Self-Learning AOA outperform both the Closest-first approach and the EHO-based approaches with a slight amelioration for DSLAOAG.

Figures 23 and 24 summarize the performances of the 7 algorithms with respect to "Travel time" on the 144 time units with 5 and 10 ambulances per base, respectively. From these figures, we notice that the three AOA-based algorithms and the three EHO-based algorithms follow the same trend, which is explained by the common strategy of updating their solutions. Also, when the number of ambulances is equal to 10, more than half of the "Travel time" scores recorded during the 144 units of time by the Algorithm Nearest (closest-first policy) vary over a considerably large interval, which denotes an inability to provide solutions in a more or less stable manner unlike the other approaches.

Figures 25 and 26 represent the statistical distribution with boxplot of Unsatisfied calls over the 144 time units. Clearly with 10 ambulances there are enough ambulances per hospital to cover all incoming calls. On the other hand, for 5 ambulances, we note that DSLAOAC represents the method which achieves the best scores in terms of the number of "Unsatisfied calls", the smallest in the long term.

As the number of ambulances = 10 yields no "Unsatisfied call" with the different compared approaches, Fig. 27 also presents no "missed Priorities". However Fig. 28 shows that Nearest algorithm does not prioritize the choice of the emergency calls and therefore obtains the worst results. Concerning the approaches based on AOA and EHO, it is clear

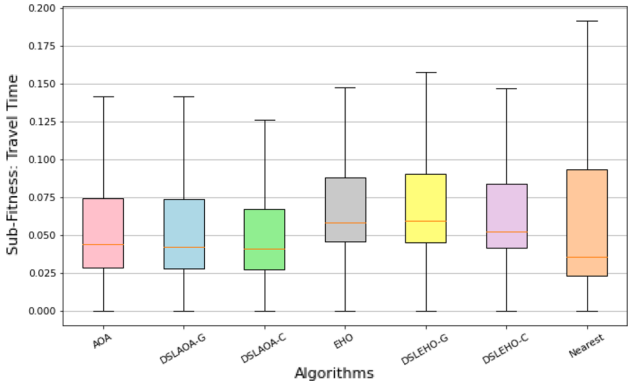


Fig. 26 Performances of the different algorithms for "Travel time" with 10 ambulances per base

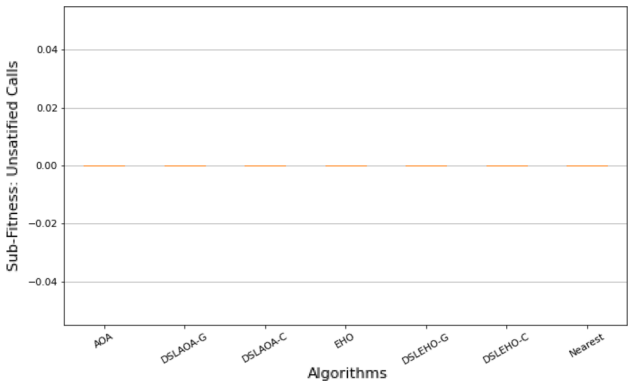


Fig. 27 Performances of the different algorithms for "Unsatisfied calls" with 10 ambulances per base

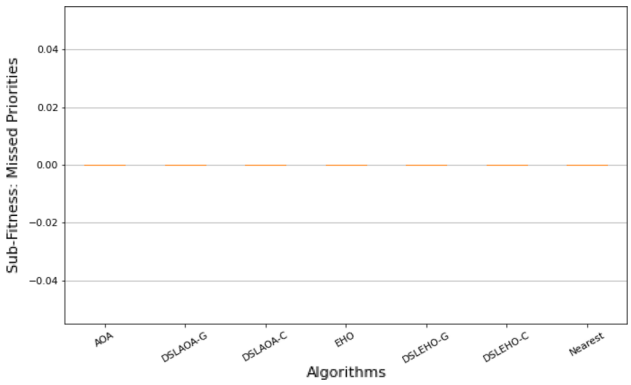


Fig. 28 Performances of the different algorithms for "missed Priorities" with 10 ambulances per base

that they take into account the impact that the priority of emergency calls can have. Nonetheless, we note that the ranges of variation of EHO, DSLEHOC, and DSLEHOG are lower than the others which means better results for this sub-fitness function.

7 Conclusion

In this paper, a quantum algorithm for the Ordering Points To Identify the Clustering Structure is designed. Its computational complexity is calculated and is shown to be reduced relatively to its classical counterpart. Experiments have been performed for an application of emergency transportation consisting in dispatching ambulances and covering urgent calls in case of Kingdom of Saudi Arabia during Covid-19 crisis. With the aim of gaining scalability for the problem solving method, clustering hospitals and clustering emergency calls have been performed in order to distribute the problem solving execution on the different clusters. First the parameters *MinPts* and *eps* were tuned by experiments for both clusterings. The obtained results helped to undertake the second part of the study very efficiently.

Afterwards, to cope with the hard issue of EMS transportation, we have proposed four novel algorithms, namely Deep Self-Learning Orcas Artificial Algorithm using Cauchy operator (DSLAOA-C), Deep Self-Learning Orcas Artificial Algorithm using Gaussian operator (DSLAOA-G), Deep Self-Learning EHO using Cauchy operator (DSLEHO-C) and Deep Self-Learning EHO using Gaussian operator (DSLEHO-G). A second set of experiments was carried out on the real life EMS transportation. The empirical parameters were first fixed by extensive experiments prior to test the performance of the proposed methods. Results proved the superiority of DSLAOA-C on both DSLAOA-G and original AOA for the three objectives. The integration of the Gaussian mutation operator has not brought improvement to original AOA as it was expected. Similar outcomes have been observed for DSLEHO-C and DSLEHO-G and original EHO. And when comparing DSLAOA-C and DSLEHO-C, we notice a better outcome for DSLAOA-C. At last, the most significant result is that all the proposed algorithms outperform the state-of-the-art algorithm namely *nearest* in the illustrated curves.

For future work, we will integrate other quantum subroutines and operators in the proposed quantum clustering algorithm in order to further improve its overall efficiency. In this work, OPTICS was experimented using Python Programming Language as it computes the same results as QOPTICS. In order to appreciate the gained calculation time in practice, we plan to code QOPTICS using Qiskit, the language proper to quantum computing that needs more investigation. Moreover, we will extend our research to develop other deep learning strategies to solve the Covid-19 EMS transportation problem. Another perspective would be to test the deep self-learning framework with larger learning depth parameter value to improve the outcomes performance. Also Quantum versions of DSLAOA and DSLEHO will be developed in order to overcome the calculation time. Another

alternative would be to code the procedures using CUDA and execute them on a GPU.

Acknowledgements We would like to express our special thanks of gratitude to Prince Mohammad Bin Fahd Center for Futuristic Studies at Prince Mohammad Bin Fahd University and the World Futures Studies Federation for the support of this work.

Author Contributions H.D.: Conceptualization of Quantum OPTICS and Deep learning for swarm intelligence, application modeling, AOA, writing and editing. Y.D.: Methodology of Quantum OPTICS and Deep Self Learning EHO, dataset construction. N.A.H.: Conceptualization and experimentation of EHO and Deep Self Learning EHO. L.S.B.: Conceptualization and experimentation of AOA and Deep Self Learning AOA, dataset construction. D.Z.: Application modeling. I.K.: Swarm intelligence background.

Funding This work was supported by Prince Mohammad Bin Fahd Center for Futuristic Studies at Prince Mohammad Bin Fahd University and the World Futures Studies Federation under the Second Futures Research Grant #13.

Data Availability The datasets generated and analysed during the current study are available at https://lria.usthb.dz/Logiciels-Datasets/IADM/Emergency_Transportation.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Bandara D, Mayorga ME, McLay LA (2014) Priority dispatching strategies for ems systems. J Oper Res Soc 65(4):572–587. https://doi.org/10.1057/jors.2013.95

Belanger V, Ruiz A, Soriano P (2019) Recent optimization models and trends in location, relocation, and dispatching of emergency medical vehicles. Eur J Oper Res 272(1):1–23. https://doi.org/10.1016/j.ejor.2018.02.055

Belanger V, Lanzarone E, Nicoletta V, Ruiz A, Soriano P (2020) A recursive simulation-optimization framework for the ambulance location and dispatching problem. Eur J Oper Res 286(2):713–725. https://doi.org/10.1016/j.ejor.2020.03.041

Bendimerad LS, Drias H (2020) An artificial orca algorithm for continuous problems. Springer, Cham, pp 700–709

Bharti K, Haug T, Vedral V, Kwek L-C (2020) Machine learning meets quantum foundations: a brief survey. AVS Quantum Sci 2(3):034101. https://doi.org/10.1116/5.0007529

Biamonte J et al (2017) Quantum machine learning. Nature 549(7671):195–202. https://doi.org/10.1038/nature23474

Boyer M, Brassard G, Høyer P, Tapp A (1998) Tight bounds on quantum searching. Fortschr Phys 46(4–5):493–505. https://doi.org/10.1002/(sici)1521-3978(199806)46:4/5493::aid-prop4933.0.co;2-p

Brassard G, Høyer P, Mosca M, Tapp A (2002) Quantum amplitude amplification and estimation. Quantum Comput Inf 2002:53–74. https://doi.org/10.1090/conm/305/05215

Carvalho A, Captivo M, Marques I (2020) Integrating the ambulance dispatching and relocation problems to maximize systems preparedness. Eur J Oper Res 283(3):1064–1080. https://doi.org/10.1016/j.ejor.2019.11.056

Castillo O, Melin P (2020) Forecasting of Covid-19 time series for countries in the world based on a hybrid approach combining the fractal dimension and fuzzy logic. Chaos Solitons Fractals 140:110242. https://doi.org/10.1016/j.chaos.2020.110242

Castillo O, Melin P (2021) A novel method for a covid-19 classification of countries based on an intelligent fuzzy fractal approach. Healthcare 9:9020196. https://doi.org/10.3390/healthcare9020196

Datasets for DADCC-COVID19 (2021). https://lria.usthb.dz/Logiciels-Datasets/IADM/Emergency_Transportation

Drias H, Drias Y, Khennak I (2021) A novel orca cultural algorithm and applications. Expert Syst J

Durr C, Heiligman M, HOyer P, Mhalla M (2006) Quantum query complexity of some graph problems. SIAM J Comput 35(6):1310–1328. https://doi.org/10.1137/050644719

Durr C, Hoyer P (1996) A quantum algorithm for finding the minimum, 92. arXiv:quant-ph/9607014. http://dx.doi.org/10.1103/PhysRevD.92.045033

Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise, KDD'96. AAAI Press, pp 226–231

Giovannetti V, Lloyd S, Maccone L (2008) Quantum random access memory. Phys Rev Lett 100(16):160501. https://doi.org/10.1103/physrevlett.100.160501

Grover LK (1996) A fast quantum mechanical algorithm for database search, STOC'96, Association for Computing Machinery, New York, NY, USA, pp 212–219. https://doi.org/10.1145/237814.237866

Houacine NA, Drias H (2021) When robots contribute to eradicate the COVID19 spread in a context of containment. Prog Artif Intell 10(4):391–416. https://doi.org/10.1007/s13748-021-00245-3

Ibri S, Nourelfath M, Drias H (2012) A multi-agent approach for integrated emergency vehicle dispatching and covering problem. Eng Appl Artif Intell 25(3):554–565. https://doi.org/10.1016/j.engappai.2011.10.003

Jakubik J, Binding A, Feuerriegel S (2021) Directed particle swarm optimization with gaussian-process-based function forecasting. arXiv:2102.04172

Kechid A, Drias H (2020) Cultural coalitions detection approach using GPU based on hybrid bat and cultural algorithms. Appl Soft Comput 93:106368. https://doi.org/10.1016/j.asoc.2020.106368

Lee S (2017) A new preparedness policy for ems logistics. Health Care Manag Sci 20:4. https://doi.org/10.1007/s10729-015-9340-4

Li J, Guo L, Li Y, Liu C (2019) Enhancing elephant herding optimization with novel individual updating strategies for large-scale optimization problems. Mathematics 7(5):395. https://doi.org/10.3390/math7050395

Li J, Lei H, Alavi AH, Wang G-G (2020) Elephant herding optimization: variants, hybrids, and applications. Mathematics 8(9):1415. https://doi.org/10.3390/math8091415

Mansour RF et al (2021) Unsupervised deep learning based variational autoencoder model for Covid-19 diagnosis and classification. Pattern Recognit Lett 151:267–274. https://doi.org/10.1016/j.patrec.2021.08.018

Melin P, Monica JC, Sanchez D, Castillo O (2020) Multiple ensemble neural network models with fuzzy response aggregation for predicting Covid-19 time series: the case of mexico. Healthcare 8(2):8020181. https://doi.org/10.3390/healthcare8020181

Mihael A, Markus MB, Hans-Peter K, Sander J (1999) Optics: ordering points to identify the clustering structure. ACM, pp 656–669

Moayedi H, Muazu MA, Foong LK (2020) Novel swarm-based approach for predicting the cooling load of residential build-

ings based on social behavior of elephant herds. Energy Build 206:109579. https://doi.org/10.1016/j.enbuild.2019.109579
Paiva FAP, Silva CRM., Leite IVO, Marcone MHF, Costa JAF (2017) Modified bat algorithm with Cauchy mutation and elite opposition-based learning, 1–6
Rami K (2021) Coronavirus. https://ramikrispin.github.io/coronavirus/
The open data portal of Saudi Arabia (2021). https://data.gov.sa/Data/en/dataset/accredited-health-service-providers_march2021
Tuba E, Capor-Hrosik R, Alihodzic A, Jovanovic R, Tuba M (2018). Chaotic elephant herding optimization algorithm IEEE. https://doi.org/10.1109/sami.2018.8324842
Usanov D, Ven P, Mei R (2019) Dispatching fire trucks under stochastic driving times. Comput Oper Res 114:104829. https://doi.org/10.1016/j.cor.2019.104829
Wang G-G, Deb S, Coelho LDS (2015) Elephant herding optimization, pp 1–5 (2015)
Wang W-C, Xu L, Xu D-M (2020) Yin-yang firefly algorithm based on dimensionally Cauchy mutation. Expert Syst Appl 150:113216. https://doi.org/10.1016/j.eswa.2020.113216
Wittek P (2014) Quantum machine learning: what quantum computing means to data mining
Zahorodko P et al (2021) Comparisons of performance between quantum-enhanced and classical machine learning algorithms on the IBM quantum experience. J Phys: Conf Ser 1840:012–021. https://doi.org/10.1088/1742-6596/1840/1/012021

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.