The Journal of Supercomputing (2022) 78:5712–5743
https://doi.org/10.1007/s11227-021-04100-z

Corona virus optimization (CVO): a novel optimization algorithm inspired from the Corona virus pandemic

Alireza Salehan[1] · Arash Deldari[1]

Accepted: 20 September 2021 / Published online: 4 October 2021

Abstract
This research introduces a new probabilistic and meta-heuristic optimization approach inspired by the Corona virus pandemic. Corona is an infection that originates from an unknown animal virus, which is of three known types and COVID-19 has been rapidly spreading since late 2019. Based on the SIR model, the virus can easily transmit from one person to several, causing an epidemic over time. Considering the characteristics and behavior of this virus, the current paper presents an optimization algorithm called Corona virus optimization (CVO) which is feasible, effective, and applicable. A set of benchmark functions evaluates the performance of this algorithm for discrete and continuous problems by comparing the results with those of other well-known optimization algorithms. The CVO algorithm aims to find suitable solutions to application problems by solving several continuous mathematical functions as well as three continuous and discrete applications. Experimental results denote that the proposed optimization method has a credible, reasonable, and acceptable performance.

Keywords Corona virus disease · Corona virus optimization · COVID-19 · SIR model · Optimization algorithms · Meta-heuristic algorithms

1 Introduction

Nowadays, due to the complexity or large number of variables in the problem space, there are many problems and applications that cannot be solved in a reasonable time and at an affordable computational cost with traditional approaches and direct search techniques. In some cases, solving the problem may require applying fundamental

✉ Alireza Salehan
salehan@torbath.ac.ir

Arash Deldari
adeldari@torbath.ac.ir

1 Department of Computer Engineering, University of Torbat Heydarieh, Torbat Heydarieh, Iran

changes to the algorithm. Therefore, finding an optimal solution to such problems is usually approached by employing meta-heuristic and evolutionary optimization algorithms. Most of these algorithms provide a convenient solution by simplifying the complex behavior of problems through the modeling of biological processes in nature. The success of these algorithms has motivated researchers to develop novel meta-heuristic and evolutionary algorithms. Accordingly, there are currently a large number of bio-inspired and nature-inspired optimization methods [1, 2].

There are different types of optimization problems that are solved by optimization algorithms: discrete, continuous, combinatorial, and multi-objective problems [3, 4]. The type of problem does not affect how algorithms are inspired by biological or natural processes. In all optimization algorithms, the performance of a process in biology or nature is examined in detail and mapped to the steps of the optimization algorithm. Due to their superior performance, optimization algorithms have been used to solve many real-world problems in various fields. Optimization algorithms can be employed in different applications, such as computer networks [5], intrusion detection [6], data mining [7], face recognition [8], and clustering [9]. Furthermore, these algorithms may be utilized in other scientific contexts like electrical engineering (i.e., power systems [10] and telecommunications [11]) and other industries (i.e., robotics [12] and transportation [13]).

In late 2019, the World Health Organization (WHO) reported cases of a new Corona virus disease in Wuhan, China [14]. The origin of this new disease was an unknown animal virus, most likely from a snake, bat, or anteater. The transmission was reportedly related to interaction among people in the Huanan seafood wholesale market [15]. In addition to its spread throughout China, the new Corona virus was transmitted globally through tourism and business travel and eventually led to a viral pandemic. Since the outbreak of the virus, known as COVID-19 (**CO**rona **VI**rus **D**isease-20**19**), millions of people around the world have become infected and even lost their lives, while others have recovered. To stop the spread of the virus, populations have adopted preventive measures and governments have implemented laws and regulations.

By considering the features and behavior of COVID-19, the current paper introduces an optimization algorithm called Corona virus optimization (CVO), which offers a practical and efficient method for solving application problems. Based on its biological and ecological characteristics, the Corona virus' epidemic behavior is simulated according to the SIR mathematical model, by which the procedure of the optimization algorithm is proposed. COVID-19 symptoms can be rapidly transmitted from one person to another and so large numbers of people in a community may become infected in very little time. With increasing human interactions over time, large populations will become infected with the virus. In addition, during this transfer, symptoms and the health condition of the person contracting the virus can differ from that of the virus carrier. The combination of the symptoms and the transmission process of the disease inspire the CVO algorithm. The proposed algorithm is applied to both minimization and maximization problems. After the theoretical development of the CVO algorithm, the present research evaluates its performance for continuous and discrete problems. This performance evaluation is conducted using a set of benchmark functions and by comparing results with other well-known optimization algorithms. The results show that the proposed algorithm's performance is acceptable and, in some cases, produces even better results than those of other existing algorithms.

To the best of our knowledge, no optimization algorithm has been proposed based on the Corona virus disease. The CVO algorithm is an optimization approach that can be of great help to researchers in solving NP-hard problems. The most important advantage of the proposed optimization algorithm is that it leads to superior results compared to most well-known optimization methods and the experimental results approve this claim. The main reason of this advantage is that the proposed method starts with an initial population, which is dynamic and the number of this population changes during repetitions, while other optimization methods consider a fixed population. This dynamic behavior denotes that only people with weak immune systems are infected which leads the population to the global best solution of the problem.

Section 2 of the paper provides the literature review. While categorizing some highly cited optimization algorithms, Sect. 2 also presents comparisons that consider some other features. Section 3 discusses the biology and ecology of the Corona virus, and Sect. 4 simulates the epidemic behavior of the Corona virus and presents the proposed optimization algorithm in detail. In Sect. 5, the performance of the proposed optimization algorithm is evaluated through the implementation of several continuous and discrete application examples. These problems consist of a set of continuous mathematical problems and applications including the optimal placement of resources, bag of tasks (BoT) scheduling, and traveling salesman problem. Finally, Sect. 6 is devoted to the conclusion, discussion, and future directions of study.

2 Related work and literature review

Biology-inspired or nature-inspired optimization algorithms consist of several different classes. Some consider the concept of swarm intelligence to model animal behavior. Algorithms in this category cover various cases, such as the locating of food by ants [16] and bees [17]; the movement of particles, including that of birds [18], bats [19], fireflies [20], grasshoppers [21], and butterflies [22]; moreover, the hunting strategies practiced by dragonflies [23], gray wolves [24], dolphins [25], and whales [26]. Other methods are inspired by the concept of reproduction-based evolution. For instance, this class considers the genetic algorithm [27], asexual reproduction optimization [28], bird mating optimization [29], coral reef optimization [30], differential evolution [31], bacterial proliferation [32], and the krill herd [33]. Other optimization algorithms have been proposed based on human social behavior, such as the imperialist competitive algorithm [34], the brain storm optimization algorithm [35], anarchic society optimization [36], and the ideology algorithm [37].

Some optimization algorithms operate on the basis of plant behavior and, in most of these, there is no interaction between the agents. For instance, this category considers invasive weed optimization [38], the plant propagation algorithm [39], the flower pollination algorithm [40], the artificial plant optimization algorithm [41], the forest optimization algorithm [42], and the tree growth algorithm [43]. Some algorithms are also based on chemical or physical processes, such as simulated annealing [44], the artificial chemical process [45], the artificial reaction algorithm [46], thermal exchange optimization [47], electromagnetism mechanism optimization [48], artificial physics optimization [49], electromagnetic field optimization [50], and the artificial electric field algorithm [51].

Other optimization approaches refer to algorithms that do not belong to any of the aforementioned categories and whose source of inspiration is a specific real-world application. Algorithms in this category include various phenomena, including meteorology, astronomy, art, and sports. Algorithms, such as the rainfall optimization algorithm [52], black hole optimization [53], the galaxy-based search algorithm [54], the pop music algorithm [55], harmony search [56], soccer game optimization [57], the golden ball algorithm [58], and FIFA world cup competitions [59], are considered in this category.

Another group of meta-heuristic algorithms based on swarm intelligence simulate the behavior of microorganisms. The operation of these algorithms is inspired by viruses, bacteria, algae, and the like. These algorithms include the virus optimization algorithm [60], virus colony search [61], bacterial-GA foraging [62], the fast bacterial swarming algorithm [63], the bacterial foraging optimization algorithm [64], bacterial colony optimization [65], and the artificial algae algorithm [66]. In virus-based bio-inspired optimization algorithms, the process of virus transmission to healthy cells is repeated until an optimal solution is obtained. For instance, in the research conducted by Juarez et al. [60], an optimization algorithm has been proposed for continuous problem that is inspired by the behavior of viruses attacking a healthy cell. In similar research by Li et al. [61], the Virus Colony Search (VCS) algorithm is proposed that simulates the virus infection and diffusion strategies. All algorithms in this category mimic the spread of a virus only in the cells of a living organism. Until now, no algorithm has been proposed that is inspired by the transmission of a virus from one individual to another.

In addition to the issues mentioned here, other optimization algorithms have been proposed, all of which are bio-inspired or nature-inspired and beyond the scope of this research. More details about these algorithms can be found in the literature [67–69]. Despite the diversity in the characteristics and behavior of all existing optimization algorithms, it can be claimed that most are based on one of the five well-known classic algorithms: ant colony optimization [16], the artificial bee colony [17], particle swarm optimization [18], the genetic algorithm [27], and differential evolution [31]. All of the algorithms mentioned above differ in their details and behavior. Table 1 compares these algorithms based on some features and characteristics.

In Table 1, the source of inspiration field shows the origin of the algorithm's concept. The solution classification field, regardless of the source of inspiration, denotes how the members of a population (algorithm solutions) interact with each other to reach the optimal solution. Stigmergy is a mechanism of indirect coordination between the different members of a population, which is performed through an intermediate structure and whose purpose is to produce better solutions [16]. In the representative solution, a number of the members of a population are first selected as the best solutions and then even better solutions are extracted from this small

Table 1 Comparison of optimization algorithms based on some characteristics

Algorithm	Acronym	Year	Source of inspiration	Solution classification	Behavior	Ref
Ant colony optimization	ACO	1996	Swarm intelligence	Stigmergy	Foraging	[16]
Artificial bee colony	ABC	2007	Swarm intelligence	Representative	Foraging	[17]
Particle swarm optimization	PSO	1995	Swarm intelligence	Representative	Movement	[18]
Bat-inspired Algorithm	BIA	2010	Swarm intelligence	Representative	Foraging	[19]
Firefly algorithm	FA	2009	Swarm intelligence	Entire population	Foraging	[20]
Grasshopper optimization algorithm	GOA	2017	Swarm intelligence	Representative	Foraging	[21]
Butterfly optimizer	BO	2015	Swarm intelligence	Representative	Movement	[22]
Dragonfly algorithm	DA	2016	Swarm intelligence	Representative	Foraging	[23]
Grey wolf optimizer	GWO	2014	Swarm intelligence	Representative	Foraging	[24]
Dolphin partner optimization	DPO	2009	Swarm intelligence	Representative	Movement	[25]
Whale optimization algorithm	WOA	2016	Swarm intelligence	Representative	Foraging	[26]
Genetic algorithm	GA	1996	Reproduction-based	Combination	Breeding	[27]
Asexual reproduction optimization	ARO	2010	Reproduction-based	Combination	Breeding	[28]
Bird mating optimization	BMO	2014	Reproduction-based	Combination	Breeding	[29]
Coral reef optimization	CRO	2014	Reproduction-based	Combination	Breeding	[30]
Differential evolution	DE	1997	Reproduction-based	Representative	Breeding	[31]
Biomimicry of social foraging bacteria for distributed optimization	BSFBDO	2002	Reproduction-based	Neighborhood	Foraging	[32]
Krill herd	KH	2012	Reproduction-based	Representative	Foraging	[33]
Imperialist competitive algorithm	ICA	2007	Social behavior	Representative	Cooperating	[34]
Brain storm optimization algorithm	BSOA	2011	Social behavior	Representative	Cooperating	[35]
Anarchic society optimization	ASO	2012	Social behavior	Representative	Not cooperating	[36]
Ideology algorithm	IA	2017	Social behavior	Representative	Searching	[37]
Invasive weed optimization	IWO	2006	Plant-based	Combination	Spreading	[38]
Plant propagation algorithm	PPA	2014	Plant-based	Representative	Spreading	[39]
Flower pollination algorithm	FPA	2012	Plant-based	Representative	Spreading	[40]

Table 1 (continued)

Algorithm	Acronym	Year	Source of inspiration	Solution classification	Behavior	Ref
Artificial plants optimization algorithm	APOA	2011	Plant-based	Entire population	Spreading	[41]
Forest optimization algorithm	FOA	2014	Plant-based	Combination	Breeding	[42]
Tree growth algorithm	TGA	2018	Plant-based	Combination	Growing	[43]
Simulated annealing	SA	1989	Chemical/physical	Combination	Tempering	[44]
Artificial chemical process	ACP	2005	Chemical/physical	Subpopulation	Reacting	[45]
Artificial reaction algorithm	ARA	2013	Chemical/physical	Combination	Reacting	[46]
Thermal exchange optimization	TEO	2017	Chemical/physical	Subpopulation	Heating	[47]
Electromagnetism mechanism optimization	EMO	2003	Chemical/physical	Entire population	Movement	[48]
Artificial physics optimization	APO	2009	Chemical/physical	Subpopulation	Movement	[49]
Electromagnetic field optimization	EFO	2016	Chemical/physical	Combination	Movement	[50]
Artificial electric field algorithm	AEFA	2019	Chemical/physical	Entire population	Movement	[51]
Rainfall optimization algorithm	RFOA	2017	Meteorology	Combination	Ranking	[52]
Black hole	BH	2013	Astronomy	Representative	Movement	[53]
Galaxy-based search algorithm	GBSA	2011	Astronomy	Combination	Movement	[54]
Pop music algorithm	PopMusic	2002	Art	Combination	Segmentation	[55]
Harmony search	HS	2005	Art	Combination	Sorting	[56]
Soccer game optimization	SGO	2012	Sport	Representative	Movement	[57]
Golden ball algorithm	GBA	2014	Sport	Combination	Team work	[58]
FIFA world cup competitions	FIFAAO	2016	Sport	Representative	Team work	[59]
Virus optimization algorithm	VOA	2009	Microorganisms	Combination	Movement	[60]
Virus colony search	VCS	2016	Microorganisms	Representative	Movement	[61]
Bacterial-GA foraging	BGAF	2007	Microorganisms	Combination	Foraging	[62]
Fast bacterial swarming algorithm	FBSA	2008	Microorganisms	Representative	Foraging	[63]
Bacterial foraging optimization algorithm	BFOA	2009	Microorganisms	Neighborhood	Foraging	[64]

Table 1 (continued)

Algorithm	Acronym	Year	Source of inspiration	Solution classification	Behavior	Ref
Bacterial colony optimization	BCO	2012	Microorganisms	Representative	Foraging	[65]
Artificial algae algorithm	AAA	2015	Microorganisms	Representative	Movement	[66]

group [18]. In the entire population method, all members of a population are effective in determining the optimal solution. In this approach, according to the difference in fitness function or the distance between solutions, a weight is assigned to each member of the population and, at each stage, this weight is used to determine the optimal solution [20]. The combination method is the crossover of several solutions with each other so as to create a better solution [27]. In a neighborhood, each solution is affected only by the solutions that are locally adjacent [32]. In the subpopulation approach, the whole population is first divided into several subpopulations. Then each solution in a subpopulation is affected only by other solutions in the same subpopulation and this process continues until the final solution is determined [45]. Moreover, the behavior field denotes the functioning procedure of the desired algorithm.

3 Corona virus biology and ecology

Since COVID-19 was first reported on December 31, 2019 in Wuhan, China, it has rapidly spread. The virus has infected people around the world and can, therefore, be considered as a pandemic [70]. As COVID-19 continues its global proliferation, governments have implemented regulations and instructed their citizens to adopt certain measures to combat the virus. Curfews, city quarantines, fines, and business closures have been deemed necessary by governments, while individuals are advised of preventive practices, such as regular hand washing, wearing masks, staying at home, and observing social distancing. Health organizations, including the WHO, have also taken steps to curb and reduce the prevalence of COVID-19. One of the most effective measures for controlling any contagious disease is its mathematical modeling. A well-designed model of an infectious disease can help predict the disease's behavior and so facilitate the planning of defensive strategies [71].

One of the most important parameters in infectious disease modeling is the basic reproductive number (denoted as R_0), which is a function of the contact rate between individuals, symptom transition probability, and the duration of infectiousness [71]. This number denotes the average number of persons infected by a single individual. If $R_0 < 1$, then each infected person can only infect less than one other person; therefore, the growth of the disease is expected to stop. If $R_0 = 1$, then each infected person can infect one other person on average, thus resulting in stable disease growth. This condition is called endemic, when the number of infected people does not increase or decrease. However, if $R_0 > 1$, then each person can infect more than one other person on average and, as a result, the disease, if left untreated, is expected to grow exponentially and consequently lead to an epidemic or pandemic [72].

Since COVID-19 is pandemic [70], the R_0 rate is certainly greater than 1. According to research, the R_0 rate for COVID-19 depends on a number of factors that include climate, race, population density, age, percentage of public transportation use, and even average income [73]. Accordingly, this number varies among different countries and even in different regions of a country. The average range for the COVID-19 R_0 rate is estimated between 2 and 12 for different parts of the world [73, 74]. Based on R_0, Fig. 1 shows how the COVID-19 virus has been transmitted and

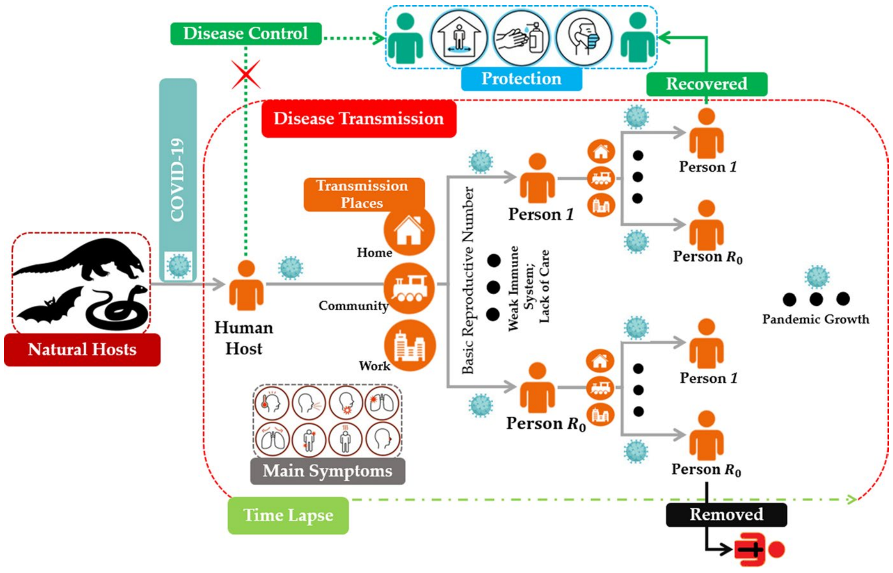


Fig. 1 How the COVID-19 virus has been transmitted and spread since its discovery, based on the value of R_0

spread since its discovery. As seen in Fig. 1, the degree of attention paid to COVID-19 protocols by the public has so far prevented the entire world population from contracting the virus. As a result, only a part of the world's population has become infected, which is perhaps due to a lack of care in following protocols or a weakened immune system.

Mathematical models of the disease can simulate the transmission process at different levels. Some types of models show cell interactions in a single patient, while others present the prevalence of the virus among different communities that are geographically dispersed. All the mathematical models proposed for COVID-19 have been of the latter type. Since the beginning of the COVID-19 outbreak, a number of mathematical models have been introduced [75–77]. The basis of most of these models is SIR [71, 78], which is a standard compartmental disease model that consists of three compartments: susceptible, infectious, and recovered/removed. One of these compartments is assigned to each member of the community.

In the SIR model, the compartment term of *susceptible* refers to those people having no immunity against the disease. Immunity can be acquired through vaccination, prior exposure, or a genetic mutation that leads to body resistance. In the model, if a susceptible individual comes in contact with an infected individual and his/her immune system is weaker than that of the infected individual, he or she becomes ill and is transferred to the infectious compartment. The compartment term of *infectious* refers to people who have the disease and can spread it to others. If an infected person is cured of the disease or he or she dies, then that person is transferred to the recovered/removed compartment. The compartment term of *recovered/removed* refers to those who have either not been infected for a long time (for reasons such as vaccination, previous exposure or high body resistance) or those who have died of a previous infection. It should be noted that, in many SIR-based models, it is assumed that a recovered/removed individual can no longer be transferred to the infectious compartment. Figure 2 illustrates the interactions in the simplest SIR model among the three COVID-19 compartments: susceptible, infectious, and recovered/removed.

If the total population is N, this population can be divided into three compartments: susceptible, infectious, and recovered/removed. Figure 2 assumes that the number of susceptible individuals is S, the number of infectious individuals is I, and the number of recovered/removed individuals is R. Accordingly, the birth rate is the percentage of people who are added to the total population of N. Moreover, the recovered/removed rate refers to the percentage of people who are transferred from the infectious to the recovered/removed compartment. The transmission rate, denoted by τ, is the rate at which symptoms are transmitted from an infected person to a susceptible person, which is directly related to the individual's extent of exposure. The dotted line above the transmission rate indicates that contact with infected individuals and the transmission of symptoms is essential for a susceptible individual to become infected. The transmission rate is calculated using Eq. 1 as follows [78]:

$$\tau = \rho \times \varphi \tag{1}$$

In Eq. 1, ρ refers to the contact rate and φ denotes the symptom transition probability. The contact rate (which is a number between 0 and 1) indicates the contact required between an infected person and a susceptible person for the transmission of symptoms. The closer this number is to zero, the less likely a person becomes infected; the closer the value is to one, the more likely a person contracts the disease. For example, $\rho = 0.5$ signifies that, for every 100 contacts, symptoms can be transmitted through 50 contacts, which then leads to illness. The symptom transition probability is between 0 and 1 and indicates the chances that an infected person transmits symptoms to a susceptible person following contact. The lower the number, the less likely disease symptoms are transmitted and, as a result, the more likely a susceptible person is not infected. However, the significant factor in Fig. 2 is the force of infection, which represents the number of people being transferred from the susceptible compartment to the infectious compartment. The force of infection, denoted as λ, is represented by members of the infectious population and is calculated as follows [78]:

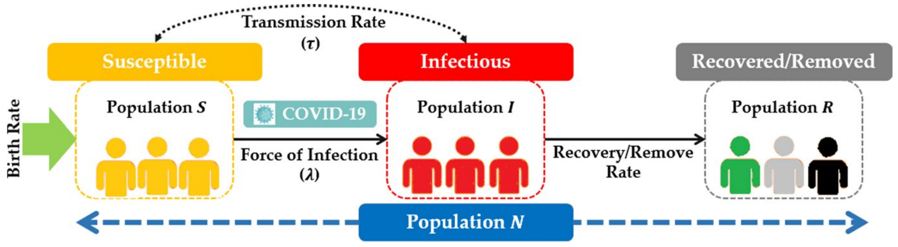


Fig. 2 The interactions in the simplest SIR model among the three COVID-19 compartments: susceptible, infectious, and recovered/removed

$$\lambda(I) = \tau \times \frac{I}{N} \tag{2}$$

Equation 1 shows that, by assuming a constant rate of symptom transmission from an infected individual to a susceptible individual, the force of infection will increase if the number of infected people increases relative to the total population and vice versa. Moreover, if the number of infected people is constant, the force of infection will increase with a rise in the symptom transmission rate and vice versa. In the SIR model, although the symptom transmission rate and the number of infected persons are critical factors in the transmission of COVID-19, the human immune system also impacts the force of infection. Accordingly, studies denote that people with a weak immune system are more susceptible to the virus [79, 80].

4 Simulating the pandemic behavior of the Corona virus infection

Based on the Corona virus' biological and ecological implications described in the previous section, Sect. 4 focuses on the pandemic behavior simulation of Corona virus transmission and introduces a new optimization method called Corona virus optimization (CVO). As observed during the Corona virus pandemic, an initial population becomes infected and, over time, the number of infected people increases according to the basic reproductive number. During the illness, symptoms can be transferred from an infected person to a susceptible person who can become infected if his or her immune system is weak. In addition, the force of infection has a significant effect on the transmission of the virus. Infected people recover or die after the course of the disease. The disease will continue to spread until either a fundamental solution to treat and stop the virus spread is found (such as a vaccine or a disruption of the transmission chain) or the immune system becomes so strong that the virus cannot be transmitted from currently infected individuals to new susceptible individuals.

The aim of the proposed method is to provide an optimal solution to optimization problems based on the pandemic behavior of COVID-19. In the proposed method, the optimization problem variables are a set of disease symptoms and the fitness function indicates the level of immunity of the susceptible person after the transmission of symptoms from an infected person. The CVO method is a representative solution-based method, meaning that it first considers an initial population infected with the virus and this population increases over time. It is important to note that, in order to achieve the optimal solution, it is better to first consider a small initial population. It is also assumed that the number of infected people in the whole virus transmission process has a maximum value (total population). This assumption is based on the fact that, despite the spread of the virus, measures taken by the public and regulations set by governments have prevented all members of the society from becoming infected with the virus. In the SIR model, although the contact rate may be between 0 and 1 for the transmission of the virus from an infected person to a susceptible person, the contact rate is assumed to be 1 due to the given force of infection of COVID-19 and the possibility of transmission even from the first contact. Moreover, the symptom transmission probability in this approach is calculated based on the type of problem (continuous or discrete) and is determined using a random number, as further explained in Sect. 4.2.

Similar to other optimization algorithms, the proposed CVO approach has a number of iterations, each of which tries to identify individuals with weaker immune systems. In each iteration, an infected person transmits symptoms to susceptible persons, the number of which is determined according to the basic reproductive number. After the transmission of symptoms to a susceptible person, his or her immune level is calculated. If the level of immunity is less than or equal to that of the infected person, then the susceptible person will also become infected. Therefore, in each iteration, an infected person can infect new susceptible individuals up to the basic reproductive number if their immunity level is less than or equal to that of the infected person. This process is also performed on other infected individuals during an iteration, and finally, new infectious populations are identified.

During each iteration, the lowest level of immunity of newly infected individuals is considered as the optimal solution in that iteration. If this value is less than the value specified in the previous iteration, the newly infected individuals will completely replace the previous infectious population. Accordingly, previous infectious populations are considered as recovered/removed populations. However, if this value is the same in two consecutive iterations, the new population is added to the previous population. Due to preventive measures adopted by the public and regulations implemented by governments, a maximum value is assumed for the infectious population in the whole transmission process. At the end of each iteration, if the total infectious population exceeds the maximum population value during the current iteration, only infected individuals with lower immunity levels are isolated and other individuals are removed and considered as recovered/removed.

4.1 The relationship between symptom transmission probability and the type of optimization problem

The transmission of COVID-19 symptoms is achieved in different ways [81]. In one transmission method, all the symptoms are transmitted from the infected individual to the susceptible individual, with the critical issue being any changes in the severity of the symptoms during transmission. For example, suppose an infected person has three symptoms: cough, fever, and the loss of taste and smell. The severity of these symptoms may increase or decrease during the transmission. In other words, all three symptoms are transmitted simultaneously and continuously to the susceptible person. However, although the susceptible person exhibits these symptoms, their intensity may differ from what the infected person experiences. In another transmission mode, the severity of the symptoms of the susceptible person may be equal to that of the infected person, but the order of their occurrence may differ. For example, consider the same infected person with the three symptoms of cough, fever, and the loss of taste and smell. During the transmission of symptoms, the susceptible person may first develop a fever, then a loss of taste and smell, and finally a cough, while the infected person first experiences a cough, then a fever followed by the loss of taste and smell. In other words, the symptoms are transmitted asynchronously and discretely to the susceptible person, who happens to experience the same intensity but a different order of occurrence.

In consideration of the different ways of transmission and the behavior of the CVO method derived from COVID-19, the symptom transmission probability depends on the type of optimization problem. In other words, during transmission, if the problem is continuous, then the severity of the symptoms changes and, if the problem is discrete, then the order of the symptoms alters. If the optimization problem is continuous, the symptom transmission probability follows a normal distribution between 0 and 1 and employs the Box-Muller transformation method. The Box-Muller transformation method, represented as Eq. 3, maps a uniform distribution in the range 0 to 1 to a standard normal distribution. In Eq. 3, variables U_1 and U_2 are two independent random numbers that are selected from a uniform distribution in a unit interval (0, 1).

$$Z = \sqrt{-2 \ln U_1} \times \cos\left(2\pi U_2\right) \tag{3}$$

By considering this policy for symptom transmission, each symptom of the infected person is randomly changed, while all of the symptoms are transmitted to the susceptible person at the same time. In other words, in continuous problems, it is assumed that all symptoms of the disease are transmitted from the infected individual to the susceptible individual with the only difference being a change in the severity of these symptoms. On the other hand, if the optimization problem is discrete, then there is a possibility of transmitting a random permutation of the symptoms from the infected person to the susceptible person. In other words, only the symptom transmission sequence changes and the severity of the symptoms is constant. In discrete problems, it is assumed that all symptoms of a disease are transmitted from the infected person to the susceptible person with the same severity, with the only difference being a change in the order of symptom transmission. However, there is another possible mode not considered by the present research which features differences in both severity and the order of symptom transmission. Consequently, this mode can be used to solve combinatorial optimization problems. If the severity and the order of symptom transmission do not change (COVID-19 symptoms can be transmitted in this way), then the proposed CVO method cannot determine a solution for the optimization problem.

4.2 Corona virus optimization algorithm

Table 2 presents the most important parameters and variables used in the CVO algorithm. The table lists the concepts inspired by COVID-19 and describes the parameters and variables considered in the optimization problem space. For example, the difference between the LocalBest and the GlobalBest is that the LocalBest variable stores the best solution in each iteration of the algorithm, while the GlobalBest variable stores the best solution in all iterations. In general, the CVO approach considers that the disease transmission process begins in a limited population and continues based on pandemic iterations. In each iteration and after the transmission of symptoms from infected to susceptible individuals, the susceptible individuals with weaker immune systems become ill and are added to the infectious population. It is assumed that individuals with stronger immune systems are transferred to the recovered/removed population. At the end of each iteration, if the value of the

Table 2 Some of the most important parameters and variables for CVO algorithm implementation

Name	Concept based on COVID-19	Description in Problem Space
Iters	The number of recurrences of the COVID-19 pandemic	The number of iterations
basePop	The number of initial infectious populations	Initial solutions
nPop	The maximum number of populations becoming infected	The maximum number of solutions in all iterations
nVar	The number of symptoms	The variables of optimization problem
LB	Lower bound of symptoms	The minimum value of each variable
UB	Upper bound of symptoms	The maximum value of each variable
GlobalBest	Patient with the weakest immune system during the pandemic	The best solution in all iterations
LocalBest	Patient with the weakest immune system during each recurrence	The best solution in each iteration
CVO	Severity of COVID-19 infection	Fitness function
R_0	Basic reproductive number	The number of new solutions created for each current solution
pop	All patients in all recurrences	The set of the solutions with less fitness function values in all iterations
newPop	New patients in each recurrence	New solutions in each iteration

weakest immune system, which is stored in the LocalBest variable, is less than the GlobalBest, then the GlobalBest variable is updated.

Algorithm 1 represents the mechanism of the proposed CVO method. The input parameters of this algorithm are Iters, basePop, nPop, nVar, LB, UB, and R_0. The output of the algorithm is the best solution which is returned by the GlobalBest variable. This variable consists of the symptoms (the optimization problem variables) and the value of the weakest immune system (the optimization problem solution) throughout the whole algorithm's iterations. Lines 1 to 16 initialize the GlobalBest variable and the initial population. In line 3 of the algorithm, according to the force of transmission of COVID-19 and the possibility of its transmission even from the first contact, the value of the contact rate is equal to 1. If the optimization problem is continuous, in lines 5 to 7 for each basePop member of the population, a set of random numbers in the interval [LB,UB] are generated and stored in pop as the initial symptoms of the current patient. However, if the optimization problem is discrete, in lines 8 to 10 for each member of the initial population, a random permutation of the problem variables is assigned as the initial symptoms. In line 11, the value of the fitness function is calculated for each member of the pop population and, if it is less than the GlobalBest value, it is replaced in lines 12 and 13.

The iterations of the CVO algorithm are performed in lines 17 to 53. In each iteration, first a population of new patients is created for the current iteration (line 18) and then the value of the LocalBest variable is initialized (lines 19 and 20). Lines 21 to 40 check the possibility of transmitting the disease from infected to susceptible individuals during a single iteration. Line 22 shows that any infected person can infect R_0 susceptible individuals. If the type of problem is continuous, the new potential symptoms are determined based on Eqs. 1, 2, 3 in lines 23 to 27, the value of which must be in the interval [LB, UB]. In line 25 of the algorithm, $|newPop|$ denotes the number of new patient populations in the current iteration. However, if the problem is discrete, lines 28 to 30 produce new susceptible symptoms by randomly replacing a permutation of symptoms of the previous patient. In line 31 of the algorithm, the immunity level of the new susceptible person is determined based on the symptoms and using the optimization fitness function. If the level of immunity for the susceptible person is less than the immunity level of the infected person transmitting the disease, the susceptible person will be added to the new patient population in lines 32–34. Moreover, if the immunity level of the susceptible person is less than the immunity level of the entire patient population during one iteration, the symptoms and immunity level of the susceptible person will replace LocalBest in lines 35 to 38 of the algorithm. It can be assumed that susceptible individuals with a higher immunity level are those who have followed health protocols and are, therefore, not added to the new patient population.

After the new patient population is determined during one iteration, the GlobalBest values and the members of the entire patient population are updated for this iteration in lines 41–52. In lines 41–46, if the value obtained for the LocalBest is less than the GlobalBest, the GlobalBest value is replaced by the LocalBest and all members of the new patient population in this iteration will replace the total population of pop patients. Otherwise, in line 45, new patients will be added to the pop patient set. In line 47 of the algorithm, the number of the initial population's members is updated for the next iteration which is equal to the number of pop members. If this initial population exceeds the total population (which is one of the algorithm's input parameters), the pop population is sorted in ascending order by the value of the immune system in lines 48–52. Therefore, the smaller nPop number is selected as the pop population and the value of the initial

po ed

pe

Algorithm 1 The **CVO** Mechanism

Inputs: The values of **Iters**, **basePop**, **nPop**, **nVar**, **LB**, **UB**, and $\boldsymbol{R_0}$

```
    // Initialization of GlobalBest and Initial Infectious Population
1   GlobalBest.Symptoms = NULL ;
2   GlobalBest.ImmuneSystem = $+\infty$ ;
3   $\boldsymbol{\rho} = 1$ ;
4   for (i = 1 ; i ≤ basePop ; i ++)
5      if (Problem type == Continues)
6         $pop_i$.Symptoms = LB ≤ nVar numbers of random values ≤ UB ;
7      endif
8      if (Problem type == Discrete)
9         $pop_i$.Symptoms = Random_Permutation (nVar) ;
10     endif
11     $pop_i$.ImmuneSystem = CVO ($pop_i$.Symptoms) ;
12     if ($pop_i$.ImmuneSystem < GlobalBest.ImmuneSystem)
13        GlobalBest.Symptoms = $pop_i$.Symptoms ;
14        GlobalBest.ImmuneSystem = $pop_i$.ImmuneSystem ;
15     endif
16  endfor_i
    // Iterations of COVID-19 Pandemic
17  for (iter = 1 ; iter ≤ Iters ; iter ++)
18     newPop = $\emptyset$ ;
19     LocalBest.Symptoms = NULL ;
20     LocalBest.ImmuneSystem = $+\infty$ ;
21     for (i = 1 ; i ≤ basePop ; i ++)
22        for (j = 1 ; j ≤ $R_0$; j ++)
23           if (Problem type == Continues)
24              $\boldsymbol{\varphi}$ = NormalRandom(0,1) ; // Equation 3
25              $\boldsymbol{\lambda} = \boldsymbol{\rho} \times \boldsymbol{\varphi} \times \frac{basePop + |newPop|}{nPop}$ ; // Equations 1 and 2
26              newSusceptible.Symptoms = LB ≤ $pop_i$.Symptoms + $\boldsymbol{\lambda}$ ≤ UB ;
27           endif
28           if (Problem type == Discrete)
29              newSusceptible.Symptoms = Random_Permutation ($pop_i$.Symptoms) ;
30           endif
31           newSusceptible.ImmuneSystem = CVO (newSusceptible.Symptoms) ;
32           if (newSusceptible.ImmuneSystem ≤ $pop_i$.ImmuneSystem)
33              Add newSusceptible to newPop ;
34           endif
35           if (newSusceptible.ImmuneSystem < LocalBest.ImmuneSystem)
36              LocalBest.Symptoms = newSusceptible.Symptoms ;
37              LocalBest.ImmuneSystem = newSusceptible.ImmuneSystem ;
38           endif
39        endfor_j
40     endfor_i
41     if (LocalBest.ImmuneSystem < GlobalBest.ImmuneSystem)
42        Replace GlobalBest with LocalBest ;
43        Replace all pop members with newPop members ;
44     else
45        Insert all newPop members to pop ;
46     endif
47     basePop = |pop| ;
48     if (basePop > nPop)
49        Sort pop members ASCENDING based on ImmuneSystem;
50        Select the first nPop members from pop ;
51        basePop = nPop ;
52     endif
53  endfor_iter
```

Objective output: The best solution **GlobalBest**

5 Performance evaluation

In order to evaluate the efficiency of the proposed method, the present study employs several mathematical functions for continuous problems and several practical examples for continuous and discrete problems. Moreover, for comparing results and evaluating performance, several well-known optimization methods are used, the most important of which are the genetic algorithm (GA), simulated annealing (SA), particle swarm optimization (PSO), and invasive weed optimization (IWO). All of the implementations are conducted in Visual Studio.NET 2010 with the C# programming language. Furthermore, all scenarios are performed on a personal computer with Intel 3.00 GHz Dual Core CPU, RAM 8.00 GB, and 1 TB HDD. The current work utilizes the best solution value criterion to test mathematical functions and evaluate practical examples. In the practical examples, the convergence of the method so as to reach the optimal solution as quickly as possible is also evaluated and discussed using several diagrams. Another reason for considering the optimal solution convergence criterion is the reduction in both the number of iterations required to determine the best solution and in the algorithm's execution time, which produce faster results. Therefore, the optimization algorithm can be utilized to solve real problems, especially real-time applications. Table 3 presents the parameter values for the CVO algorithm in all the evaluations. Similar to the CVO method, the total population size is 100 and the number of iterations is 200 in all other algorithms.

5.1 Continuous mathematical functions

There are many mathematical functions used to evaluate optimization algorithms. Table 4 lists the selected mathematical functions to evaluate the CVO method. These continuous functions are the most common functions for evaluating most optimization algorithms. Due to their specific input range and best global value, these mathematical functions are considered appropriate for evaluating new optimization algorithms. In addition to the function name and its mathematical representation, Table 4 lists other fields, namely the number of dimensions, the input domain, the best solution value, and the 3D schematic of function graph.

In order to accurately evaluate the proposed optimization method, a comparison with other optimization algorithms is conducted as well as a repeat of all experiments 10 times. Table 5 presents the average results for the best solution criterion. Since the number of variables affects the results of the algorithm, the number of variables in the different experiments varies. Consequently, all of the experiments for each algorithm are performed 30 times. The results in Table 5 indicate that the proposed method is

Table 3 The common parameter values of the CVO algorithm in all experiments

Parameter	Initial population size	Population size	R_0	ρ	Number of iterations
Value	10	100	5	1	200

Table 4 Continuous mathematical functions selected to evaluate the CVO method

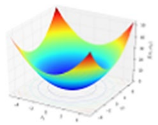
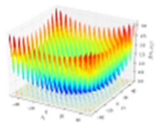
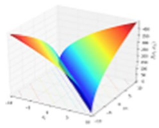
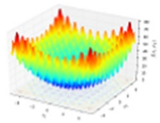
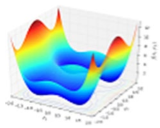
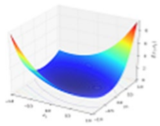
Function name	Number of dimensions	Mathematical definition	Input domain	Best solution value	3D schematic of function graph
Sphere	n	$f(x) = \sum_{i=1}^{n} x_i^2$	[− 5.12, 5.12]	0.0	
Griewank	n	$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[− 600, 600]	0.0	
ZeroSum	n	$f(x) = \begin{cases} 0 if \sum_{i=1}^{n} x_i = 0 \\ 1 + \sqrt{10000 \lvert \sum_{i=1}^{n} x_i \rvert} otherwise \end{cases}$	[− 10, 10]	0.0	
Rastrigin	n	$f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right]$	[− 5.12, 5.12]	0.0	
Qing	n	$f(x) = \sum_{i=1}^{n} (x_i^2 - i)^2$	[− 500, 500]	0.0	
Zacharov	n	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2} \sum_{i=1}^{n} ix_i\right)^2 + \left(\frac{1}{2} \sum_{i=1}^{n} ix_i\right)^4$	[− 5, 10]	0.0	

Table 4 (continued)

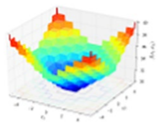
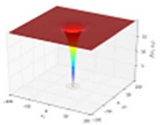
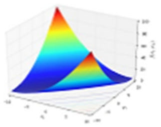
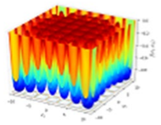
Function name	Number of dimensions	Mathematical definition	Input domain	Best solution value	3D schematic of function graph
Plateau	n	$f(x) = 30 + \sum_{i=1}^{n} x_i$	[− 5.12, 5.12]	0.0	
Easom	2	$f(x) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$	[− 100, 100]	− 1.0	
Matyas	2	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	[− 10, 10]	0.0	
PenHolder	2	$f(x) = -e^{\left\vert \left\vert e^{\left\vert -\frac{\sqrt{x_1^2+x_2^2}}{\pi}+1 \right\vert} \cos(x_1)\cos(x_2) \right\vert \right\vert^{-1}}$	[− 11, 11]	− 0.9635	

Table 5 Average results for 10 executions of benchmark mathematical functions using optimization algorithms and based on the number of different variables

Function	$f(x^*)$	Number of variables	Average results				
			GA	SA	PSO	IWO	CVO
Sphere	0.0	5	2.352E−11	1.352	6.066E−15	2.782	2.435E−11
		10	8.266E−06	7.095	7.724E−12	1.058E+01	1.668E−05
		50	0.372	9.206E+01	0.069	8.908E+01	3.067
Griewank	0.0	5	0.017	2.254	0.041	9.056	0.005
		10	0.149	2.526E+01	0.296	4.128E+01	1.111
		50	7.439	2.543E+02	1.358	3.213E+02	9.817
ZeroSum	0.0	5	1.078	2.341	1.051	1.026	1.015
		10	1.473	2.544	2.337	1.037	1.045
		50	1.814	3.207	5.099	1.053	1.385
Rastrigin	0.0	5	3.884E−05	7.634	3.262E−08	6.909	1.006E−09
		10	2.873	4.708E+01	1.985	6.192E+01	0.978
		50	3.432E+02	5.641E+03	1.728E+02	6.695E+02	6.097E+01
Qing	0.0	5	1.337E−13	1.463E+03	2.285E−11	2.138E+04	0.061
		10	0.354	5.108E+07	1.816E−05	6.642E+08	1.235
		50	1.633E+07	3.636E+11	4.272E+06	7.248E+11	5.532E+07
Zacharov	0.0	5	5.772E−07	1.787	4.477E−14	1.898E+01	1.317E−08
		10	0.068	3.196E+01	1.076E−06	5.618E+02	0.008
		50	7.483E+01	8.394E+02	1.473E+01	2.069E+05	6.428E+01
Plateau	0.0	5	0.0	8.0	0.0	1.20E+01	0.0
		10	− 2.40E+01	− 1.0	− 3.00E+01	4.0	− 3.00E+01
		50	− 1.58E+02	− 6.50E+01	− 2.48E+02	− 1.90E+01	− 2.21E+02
Easom	− 1.0	2	− 1.0	− 2.561E−55	− 1.0	− 2.715E−25	− 1.0
Matyas	0.0	2	4.403E−16	0.001	1.362E−21	0.089	3.421E−18
PenHolder	− 0.9635	2	− 2.308	− 2.310E−32	− 1.834	− 1.478E−21	− 1.029

$f(x^*)$ refers to the best solution value of the functions

able to determine a suitable solution for most of Table 4's mathematical functions for different numbers of variables and leads to acceptable solutions for other functions.

5.2 Optimal placement of resources

The problem of optimal placement of resources is considered a continuous problem that has many applications in everyday life and industries, such as electricity, computers, and mechanics. Assuming that there are a number of devices installed in the environment, the goal of this problem is to find the best resource placement with the shortest distance from all devices. Shorter distances either reduce the cost of connecting devices to the resource or increase the possibility of accessing the resource. It should be noted that, in addition to the distance criterion, there may be other important criteria depending on the type of application. Due to existing geographical constraints, it is generally not possible to use the unique shortest path to calculate the distance in this case and so the Manhattan distance is employed instead

[82]. The resource and devices can be located in an n-dimensional geographic space and, due to the fact that the geographical coordinates are continuous, this is considered as a continuous problem. Generally, finding a suitable resource placement is difficult and time consuming, especially when the number of devices and the size of the geographic space increase; as a result, this is considered as an NP-hard problem. Optimization algorithms are, therefore, a good option when finding an optimal solution. Eq. 4 presents the objective function of this problem.

$$\begin{aligned} &\min \text{ Distance} = \sum_{j=1}^{m} \sum_{i=1}^{n} \left| R_i - D_{\text{ji}} \right| \\ &s.t. : \\ &\forall i,j = 1,2,\ldots,m(i \neq j) : \sum_{k=1}^{n} \left| D_{ik} - D_{\text{jk}} \right| \neq 0 \end{aligned} \tag{4}$$

In an n-dimensional space, suppose there is a resource that must be connected to m number of devices. The location of this resource should be specified as a place where all devices are at the shortest distance from the resource. In Eq. 4, index i represents each dimension in the coordinate axes, ($i \in \{1,2,\ldots,n\}$), and index j represents the device number ($j \in \{1,2,\ldots,m\}$). Variables R_i and D_{ji} also denote the coordinates of the i-th dimension of the resource and the i-th dimension of the j-th device, respectively. Moreover, $\left|R_i - D_{ji}\right|$ represents the absolute distance of the i-th dimension of the resource and the i-th dimension of the j-th device relative to each other. Although Eq. 4 uses the Manhattan distance, other geometric distances, such as the Euclidean distance, can also be considered depending on the problem. The limitation of Eq. 4 states that independent devices are located in separate locations and this deployment independence may be considered in only one dimension. To solve the problem of the optimal placement of resources using the CVO method, each resource placement is assumed to be a patient whose symptoms are the geographical coordinates of the resource.

Figure 3 provides the experimental results, to determine the optimal solution to this problem using the CVO method in comparison with other optimization algorithms. To implement this problem, the number of devices is assumed to be 1,000 and their position is randomly determined in a five-dimensional space. Although this space is unrealistic, these features are considered so as to increase the number of variables and thus augment the complexity of the problem. The comparison criterion is the sum of the minimum distances between the resource and the devices in all five dimensions based on Eq. 4. As shown in Fig. 3, while determining the appropriate solution, the CVO method is able to converge toward this solution with the least possible iterations when compared to other algorithms.

5.3 Bag of tasks (BoT) scheduling

Bag of tasks scheduling is one of the well-known problems in the field of high-performance computing (HPC). In the context of BoT scheduling, the main goals are to

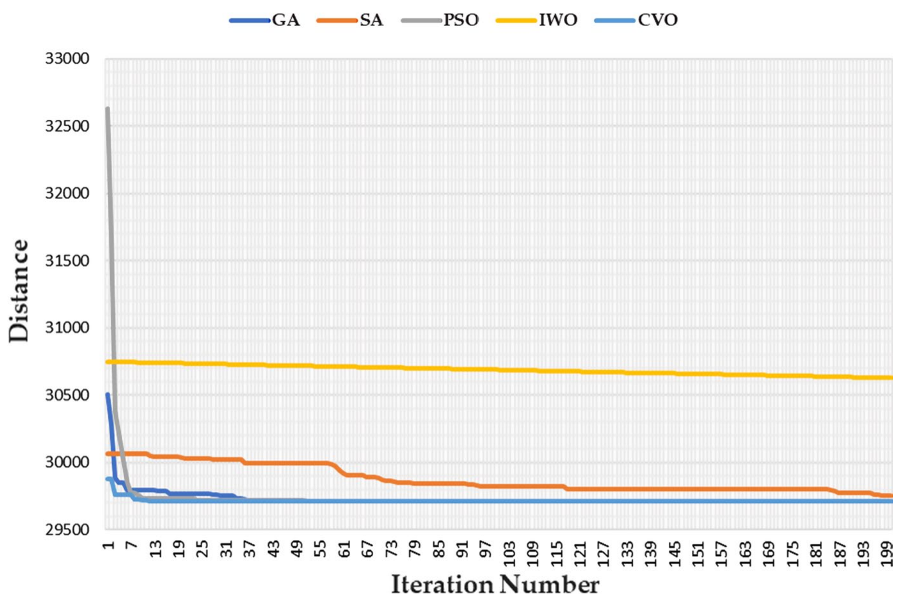


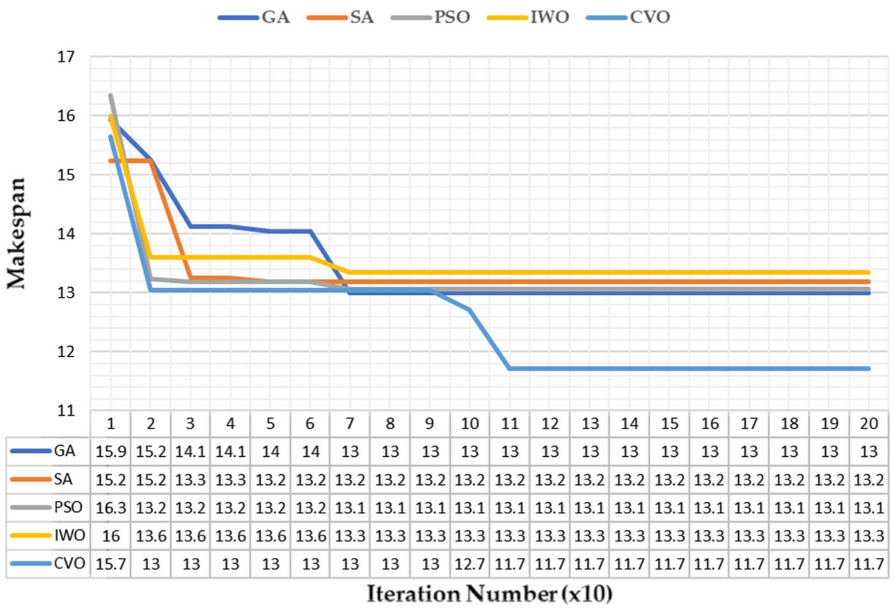
Fig. 3 Comparison of the experimental results of the problem of optimal placement of resources using the CVO method with those of other algorithms

reduce the makespan and thus increase system utilization [83]. In this discrete problem, there are a number of tasks that must be scheduled and executed on different processing resources. Because of the increase in the number and size of tasks (according to the number of instructions), finding a suitable scheduling solution is difficult and time consuming. Accordingly, the problem of BoT scheduling is an NP-hard problem that can be solved with optimization algorithms. In high-performance computing, namely cloud and grid computing, there are a number of virtual machines that differ in their configurations, such as for processing speed, memory, and storage. Moreover, tasks can have a variety of computational, memory, and storage requirements. Therefore, the cost of executing a task on different virtual machines varies. The goal is to find the best assignment of tasks to the virtual machines so that the makespan is minimized. Equation 5 expresses the objective function of this problem:

$$\begin{aligned} &\min \text{Makespan} = \sum_{i=1}^{n} \sum_{j=1}^{m} C_{\text{ij}} \times X_{\text{ij}} \\ &s.t.: \\ &X_{\text{ij}} = \begin{cases} 1 & \text{if Task}_i \text{is executed on VM}_j \\ 0 & \text{Otherwise} \end{cases} \\ &\sum_{j=1}^{m} X_{\text{ij}} = 1 \end{aligned} \tag{5}$$

Suppose there are n tasks that must be scheduled on m virtual machines. In this equation, index i represents the number of tasks ($i \in \{1,2,\ldots,n\}$) and index j represents the number of virtual machines ($j \in \{1,2,\ldots,m\}$); variable C_{ij} also represents the cost of executing the i-th task on the j-th virtual machine. The first constraint of this equation shows that if task i is executed on virtual machine j, the value of flag X_{ij} will be equal to 1 or otherwise equal to 0. The other limitation of Eq. 5 also states that, during the scheduling and execution of tasks, each task can only be executed on one virtual machine. To solve the BoT scheduling problem with the CVO method, each scheduling sequence is considered as a patient whose symptoms are the cost of executing tasks on virtual machines.

Figure 4 shows the experimental results of determining the optimal solution to the BoT scheduling problem with the CVO method in comparison with other algorithms. To implement this problem, 300 tasks and 50 virtual machines are considered. In addition, the number of instructions for each task and the processing power of each virtual machine are randomly determined. The comparison criterion is the makespan based on Eq. 5. As depicted in Fig. 4, the CVO method is able to determine the appropriate solution in the least possible iterations when compared to other algorithms. Figure 5 also shows the schedule map of the tasks on virtual machines after the implementation of the CVO method in the developed simulator.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
GA	15.9	15.2	14.1	14.1	14	14	13	13	13	13	13	13	13	13	13	13	13	13	13	13
SA	15.2	15.2	13.3	13.3	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2
PSO	16.3	13.2	13.2	13.2	13.2	13.2	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1	13.1
IWO	16	13.6	13.6	13.6	13.6	13.6	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3
CVO	15.7	13	13	13	13	13	13	13	13	12.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7

Fig. 4 Experimental results of the BoT scheduling problem using the CVO method in comparison with those of other algorithms

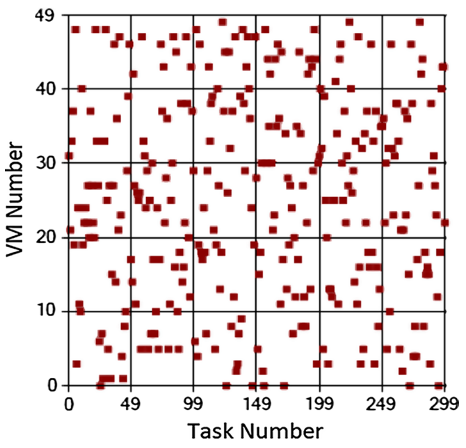
5.4 Traveling salesman problem

One of the famous problems addressed by many optimization algorithms and having numerous applications in the transportation industry is the traveling salesman problem (TSP) [84]. In this discrete problem, a traveling salesman starts marketing from one city of origin and returns to the city where he started after meeting his customers in *n* other cities. Accordingly, each city is visited only once and traveling from one city to another produces transportation costs. The goal is to find the route with the least cost for the salesman and in which all cities are traversed only once. Since it is assumed that there is a route between all the cities, the number of possible solutions to this problem is a permutation of *n*. The purpose of the problem is to find the path with the lowest transportation cost between these *n!* modes. As the number of cities increases, the number of possible solutions grows significantly. Due to this increase in problem space, the TSP is considered as an NP-hard problem. Equation 6 shows the objective function of this problem:

$$
\begin{aligned}
&\min \text{Traveling Cost} = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{\text{ij}} \times X_{\text{ij}} \\
&s.t.: \\
&X_{ij} = \begin{cases} 1 & \text{ifSellermovesfromCity}_i to \text{City}_j \\ 0 & \text{Otherwise} \end{cases} \\
&\sum_{i=1}^{n} X_{\text{ij}} = 1 \\
&\sum_{j=1}^{n} X_{\text{ij}} = 1
\end{aligned}
\tag{6}
$$

Suppose there are *n* cities that the salesman must visit. In this equation, indices *i* and *j* represent the numbers of origin and destination cities, respectively

Fig. 5 The schedule map of BoT scheduling after executing the CVO method in the developed simulator



($i,j \in \{1,2,\ldots,n\}$), and variable C_{ij} represents the cost of traveling from city i to city j. The first constraint of this equation denotes that, if the salesman travels from city i to city j along the route, the value of flag X_{ij} will be equal to 1 or otherwise equal to 0. The last two constraints of Eq. 6 also state that, along the optimal route, each city can only be visited once as an origin and once as a destination. To solve the traveling salesman problem using the CVO method, each traveled route by the salesman from all cities is considered as one patient whose symptoms are the cost of traveling between pairs of cities.

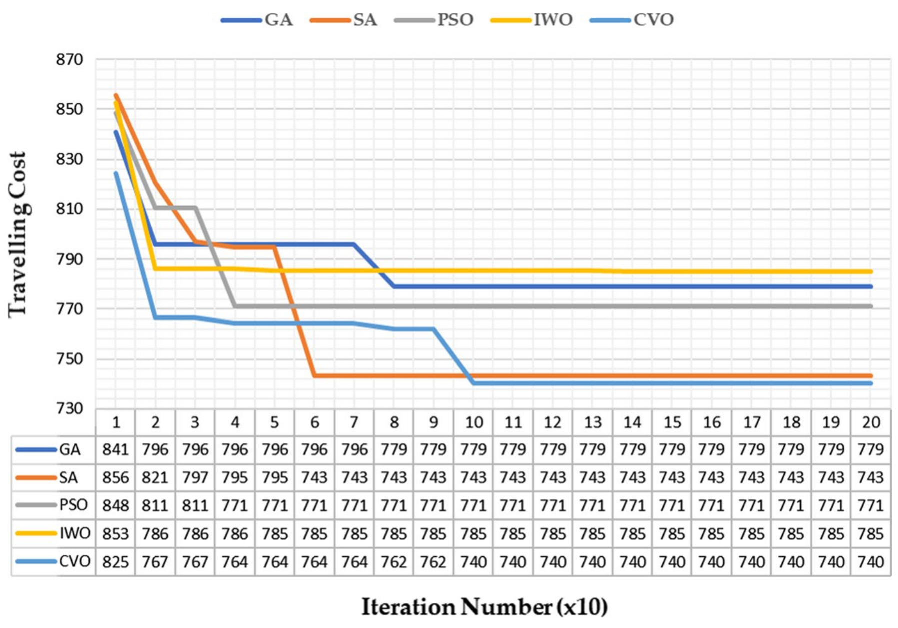
To implement the traveling salesman problem on different optimization algorithms, three types of datasets are considered. In the first set, 100 cities are produced with random distances between 0.1 and 20. In the second set, the distances between 48 central cities in different US states are used from the ATT48 dataset [85]. In the ATT48 dataset, which is dedicated to evaluating algorithms developed for the traveling salesman problem, the tour has a minimum length of 33,523. The third dataset also considers the data presented in [43] consisting of 15 cities. The minimum tour length in this dataset is 248.03. The comparison criterion is the minimum cost of traveling based on Eq. 6. Figures 6, 7, 8 show the experimental results of determining the optimal solution with the CVO method in comparison with other algorithms for the three datasets. As depicted in these figures, the CVO method, while determining the appropriate solution, is able to converge toward this solution in the least possible iterations when compared to other algorithms.

Figure 9 also presents the optimal route between cities after the implementation of the CVO method in the developed simulator for each of the selected datasets. All the results in Sects. 5.2,5.3, 5.4indicate that the CVO method is able to determine the optimal solution with the least possible iterations. In other words, the algorithm converges to the optimal solution with less iterations and at a higher speed. Accordingly, the execution time of the CVO algorithm is significantly reduced and, as a result, the proposed method can be used for various practical applications.

6 Conclusion, discussion, and future work

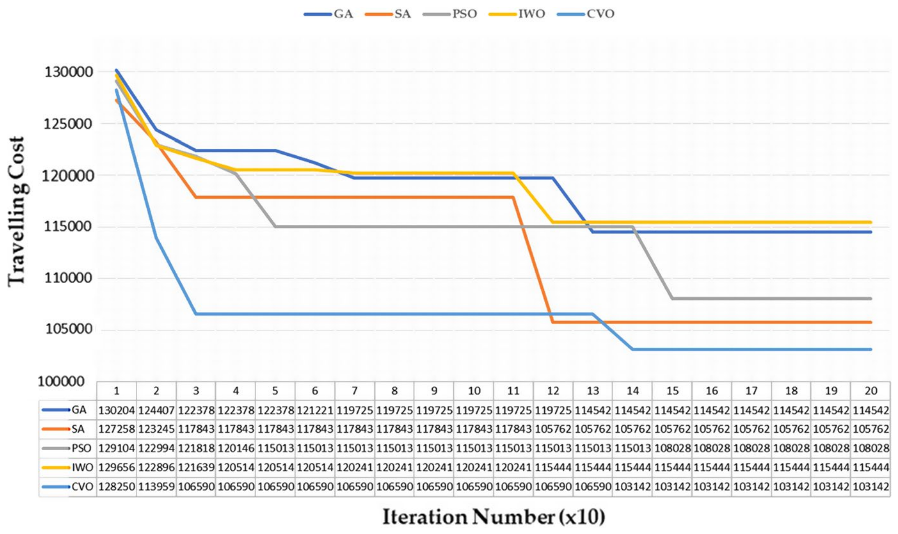
The current research simulated the pandemic behavior of Corona virus transmission based on the biological and ecological features of COVID-19 and, as a result, introduced a new optimization method called Corona virus optimization (CVO). Compared to some well-known optimization methods, such as the genetic algorithm, simulated annealing, particle swarm optimization, and invasive weed optimization, the proposed method can solve optimization problems and find appropriate solutions in a timely manner and with minimum iterations. In some cases, the solution obtained by the proposed algorithm is better than the mentioned optimization methods. To evaluate the performance of the proposed CVO method, the present study selects several continuous mathematical functions as well as three continuous and discrete application problems, namely the optimal placement of resources, bag of tasks scheduling, and traveling salesman problem.

The CVO algorithm is developed in such a way that its behavior is well-defined and, depending on the type of problem (discrete or continuous), there is no need



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
GA	841	796	796	796	796	796	796	779	779	779	779	779	779	779	779	779	779	779	779	779
SA	856	821	797	795	795	743	743	743	743	743	743	743	743	743	743	743	743	743	743	743
PSO	848	811	811	771	771	771	771	771	771	771	771	771	771	771	771	771	771	771	771	771
IWO	853	786	786	786	785	785	785	785	785	785	785	785	785	785	785	785	785	785	785	785
CVO	825	767	767	764	764	764	764	762	762	740	740	740	740	740	740	740	740	740	740	740

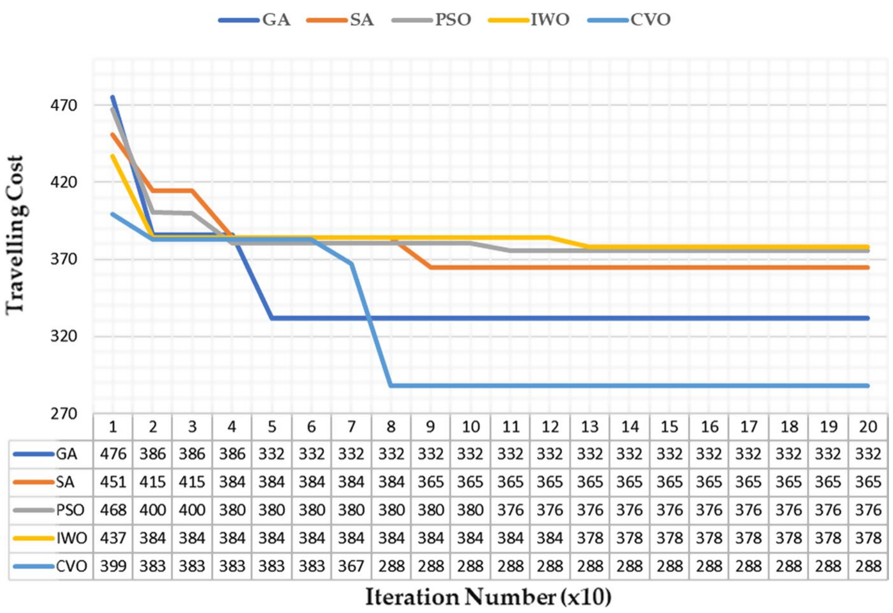
Fig. 6 Experimental results of the traveling salesman problem using a random dataset



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
GA	130204	124407	122378	122378	122378	121221	119725	119725	119725	119725	119725	119725	114542	114542	114542	114542	114542	114542	114542	114542
SA	127258	123245	117843	117843	117843	117843	117843	117843	117843	117843	117843	105762	105762	105762	105762	105762	105762	105762	105762	105762
PSO	129104	122994	121818	120146	115013	115013	115013	115013	115013	115013	115013	115013	115013	115013	108028	108028	108028	108028	108028	108028
IWO	129656	122896	121639	120514	120514	120514	120241	120241	120241	120241	120241	115444	115444	115444	115444	115444	115444	115444	115444	115444
CVO	128250	113959	106590	106590	106590	106590	106590	106590	106590	106590	106590	106590	106590	103142	103142	103142	103142	103142	103142	103142

Fig. 7 Experimental results of the traveling salesman problem using the ATT48 dataset [85]

to convert discrete problems to continuous problems and vice versa or to convert data for that matter. The transparent behavior of this algorithm is such that it can be utilized to easily develop and solve a discrete or continuous optimization problem. The experimental results indicated that, for both discrete and continuous cases, the



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
GA	476	386	386	386	332	332	332	332	332	332	332	332	332	332	332	332	332	332	332	332
SA	451	415	415	384	384	384	384	384	365	365	365	365	365	365	365	365	365	365	365	365
PSO	468	400	400	380	380	380	380	380	380	380	376	376	376	376	376	376	376	376	376	376
IWO	437	384	384	384	384	384	384	384	384	384	384	384	378	378	378	378	378	378	378	378
CVO	399	383	383	383	383	383	367	288	288	288	288	288	288	288	288	288	288	288	288	288

Fig. 8 Experimental results of the traveling salesman problem using the dataset presented in [43]

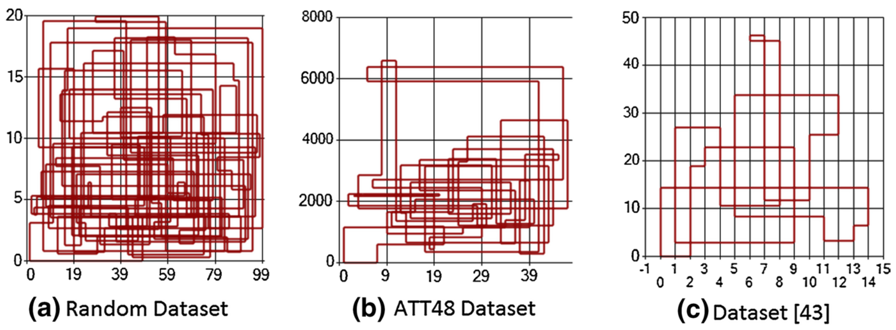


Fig. 9 The optimal route between cities after the implementation of the CVO method for each of the selected datasets: **a** random dataset; **b** ATT48 dataset; **c** dataset in [43]

proposed method provides a credible, acceptable, and reasonable performance. To solve a problem and find the best answer in most existing optimization methods, the algorithm settings must be determined by trial-and-error and be based on the type of problem. However, the CVO method does not require a continuous change of parameters. All evaluations show that it is possible to solve the optimization problem and find the optimal solution with the same default values for the algorithm, all of which are based on COVID-19 pandemic behavior.

In the standard SIR model, it is assumed that infected individuals will not fall ill again after their transfer to the recovered/removed compartment and so will not be transferred to the susceptible and infectious compartments. This assumption is not the case with COVID-19 since people who have already been infected are still susceptible to the disease and may re-enter the susceptible compartment. Despite this, the proposed method follows the standard SIR model. The reason for this behavior is because, in each iteration of the algorithm, the value of the fitness function for new solutions (susceptible individuals) is less than that of the previous solution (infected individuals). Therefore, in each iteration, there are infectious populations that have the lowest value of the fitness function when compared to all previous infectious populations. In other words, over time, populations are selected that are new and the output value of their fitness function is less than those of all the solutions obtained so far.

Although this research presents the CVO method for solving minimization problems and all experiments are performed accordingly, this approach may also be employed to solve maximization problems. To do so, an incremental precondition, such as the mutated strength of the virus instead of a weak immune system, is considered when selecting the output of the fitness function so that, in each iteration of the algorithm, higher value solutions are selected instead of the previous solutions. On the other hand, the value of the basic reproductive number can affect the performance of the CVO method. If the value of R_0 is considered too low, the problem space is reduced and the final solution may not be the optimal solution. Moreover, if the value of R_0 is considered too large, the pandemic will rapidly spread and the number of infectious populations will increase. This assumption will affect the execution time of the algorithm in achieving the optimal solution. According to the various experiments performed, the value of $4 \leq R_0 \leq 6$ can be a reasonable selection, in which the value of R_0 is assumed to be 5 in the current research.

As for future directions, more focus will be placed on the values of CVO algorithm parameters and their effect on how the algorithm functions. Moreover, the performance of the proposed CVO algorithm can be improved when inspired by various COVID-19 mutations, such as the UK variant. In addition, the CVO method may be utilized for other applications, especially real-time, and its performance evaluated. Another future direction is to modify the CVO algorithm for use in solving combinatorial optimization problems. In this regard, the usage will be for both types of symptom transmissions: a change in the severity of symptoms for continuous problems and a change in the order of symptoms transmitted for discrete problems.

References

1. Yang XS, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (2013) Swarm intelligence and bio-inspired computation: theory and applications. Newnes, Elsevier, London. https://doi.org/10.1016/B978-0-12-405163-8.00020-X
2. Yang XS (2014) Nature-inspired optimization algorithms. Academic Press, Elsevier, London. https://doi.org/10.1016/C2013-0-01368-0
3. Zavala GR, Nebro AJ, Luna F, Coello CAC (2014) A survey of multi-objective metaheuristics applied to structural optimization. Struct Multidiscip Optim 49(4):537–558. https://doi.org/10.1007/s00158-013-0996-4
4. Molina D, LaTorre A, Herrera F (2018) An insight into bio-inspired and evolutionary algorithms for global optimization: review, analysis, and lessons learnt over a decade of competitions. Cogn Comput 10(4):517–544. https://doi.org/10.1007/s12559-018-9554-0

5. Dressler F, Akan OB (2010) A survey on bio-inspired networking. Comput Netw 54(6):881–900. https://doi.org/10.1016/j.comnet.2009.10.024
6. Kolias C, Kambourakis G, Maragoudakis M (2011) Swarm intelligence in intrusion detection: a survey. Comput Secur 30(8):625–642. https://doi.org/10.1016/j.cose.2011.08.009
7. Fong S (2013) Opportunities and challenges of integrating bio-inspired optimization and data mining algorithms. In: Yang XS, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (eds) Swarm intelligence and bio-inspired computation: theory and applications. Newnes, Elsevier, London, pp 385–402. https://doi.org/10.1016/B978-0-12-405163-8.00018-1
8. Alsalibi B, Venkat I, Subramanian K, Lutfi SL, Wilde PD (2015) The impact of bio-inspired approaches toward the advancement of face recognition. ACM Comput Surv (CSUR) 48(1):1–33. https://doi.org/10.1145/2791121
9. Jose-Garcia A, Gomez-Flores W (2016) Automatic clustering using nature-inspired metaheuristics: a survey. Appl Soft Comput 41:192–213. https://doi.org/10.1016/j.asoc.2015.12.001
10. Del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG (2008) Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans Evol Comput 12(2):171–195. https://doi.org/10.1109/TEVC.2007.896686
11. Yang XS, Chien SF, Ting TO (2015) Bio-inspired computation and optimization: an overview. In: Yang XS, Chien SF, Ting TO (eds) Bio-inspired computation in telecommunications. Morgan Kaufmann, Elsevier, pp 1–21. https://doi.org/10.1016/B978-0-12-801538-4.00001-X
12. Beni G, Wang J (1993) Swarm intelligence in cellular robotic systems. In: Dario P, Sandini G, Aebischer P (eds) Robots and biological systems: Towards a new bionics? Springer, Berlin, Heidelberg, pp 703–712. https://doi.org/10.1007/978-3-642-58069-7_38
13. Del Ser J, Osaba E, Sanchez-Medina JJ, Fister I (2019) Bioinspired computational intelligence and transportation systems: a long road ahead. IEEE Trans Intell Transp Syst 21(2):466–495. https://doi.org/10.1109/TITS.2019.2897377
14. World Health Organization (2020) Coronavirus. World Health Organization. https://www.who.int/health-topics/coronavirus. Accessed 19 May 2020
15. Li Q, Guan X, Wu P, Wang X, Zhou L, Tong Y, Ren R, Leung KS, Lau EH, Wong JY, Xing X (2020) Early transmission dynamics in Wuhan, China, of novel coronavirus–infected pneumonia. N Engl J Med 382:1199–1207. https://doi.org/10.1056/NEJMoa2001316
16. Dorigo M, Maniezzo V, Colorni A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern 26(1):29–41
17. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471. https://doi.org/10.1007/s10898-007-9149-x
18. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: *MHS'95, the sixth international IEEE symposium on micro machine and human science*, pp 39–43. https://doi.org/10.1109/MHS.1995.494215
19. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Gonzalez JR, Sancho-Royo A, Pelta DA, Cruz C (eds) Nature-inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, Heidelberg, pp 65–74. https://doi.org/10.1007/978-3-642-12538-6_6
20. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Watanabe O, Zeugmann T (eds) Stochastic algorithms: foundations and applications. Springer, Berlin, Heidelberg, pp 169–178. https://doi.org/10.1007/978-3-642-04944-6_14
21. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004
22. Kumar A, Misra RK, Singh D (2015) Butterfly optimizer. In: *2015 IEEE workshop on computational intelligence: theories, applications and future directions (WCI)*. Kanpur, India, p. 1–6. https://doi.org/10.1109/WCI.2015.7495523
23. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073. https://doi.org/10.1007/s00521-015-1920-1
24. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
25. Shiqin Y, Jianjun J, Guangxing Y (2009) A dolphin partner optimization. In: *2009 WRI global congress on intelligent systems (IEEE)*. Xiamen, China, pp 124–128. https://doi.org/10.1109/GCIS.2009.464

26. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
27. Man KF, Tang KS, Kwong S (1996) Genetic algorithms: concepts and applications [in engineering design]. IEEE Trans Industr Electron 43(5):519–534. https://doi.org/10.1109/41.538609
28. Farasat A, Menhaj MB, Mansouri T, Moghadam MRS (2010) ARO: a new model-free optimization algorithm inspired from asexual reproduction. Appl Soft Comput 10(4):1284–1292. https://doi.org/10.1016/j.asoc.2010.05.011
29. Askarzadeh A (2014) Bird mating optimizer: an optimization algorithm inspired by bird mating strategies. Commun Nonlinear Sci Numer Simul 19(4):1213–1228. https://doi.org/10.1016/j.cnsns.2013.08.027
30. Salcedo-Sanz S, Del Ser J, Landa-Torres I, Gil-Lopez S, Portilla-Figueras JA (2014) The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems. Sci World J. https://doi.org/10.1155/2014/739768
31. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359. https://doi.org/10.1023/A:1008202821328
32. Liu Y, Passino KM (2002) Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors. J Optim Theory Appl 115(3):603–628. https://doi.org/10.1023/A:1021207331209
33. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845. https://doi.org/10.1016/j.cnsns.2012.05.010
34. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: *2007 IEEE congress on evolutionary computation*. Singapore, pp 4661–4667. https://doi.org/10.1109/CEC.2007.4425083
35. Shi Y (2011) Brain storm optimization algorithm. In: Tan Y, Shi Y, Chai Y, Wang G (eds) Advances in swarm intelligence (ICSI 2011) lecture notes in computer science. Springer, Berlin, Heidelberg, pp 303–309
36. Shayeghi H, Dadashpour J (2012) Anarchic society optimization based PID control of an automatic voltage regulator (AVR) system. Electr Electron Eng 2(4):199–207. https://doi.org/10.5923/j.eee.20120204.05
37. Huan TT, Kulkarni AJ, Kanesan J, Huang CJ, Abraham A (2017) Ideology algorithm: a socio-inspired optimization methodology. Neural Comput Appl 28(1):845–876. https://doi.org/10.1007/s00521-016-2379-4
38. Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Eco Inform 1(4):355–366. https://doi.org/10.1016/j.ecoinf.2006.07.003
39. Sulaiman M, Salhi A, Selamoglu BI, Kirikchi OB (2014) A plant propagation algorithm for constrained engineering optimisation problems. Math Probl Eng. https://doi.org/10.1155/2014/627416
40. Yang XS (2012) Flower pollination algorithm for global optimization. In: Durand-Lose J, Jonoska N (eds) Unconventional computation and natural computation (UCNC 2012) lecture notes in computer science. Springer, Berlin, Heidelberg, pp 240–249. https://doi.org/10.1007/978-3-642-32894-7_27
41. Zhao Z, Cui Z, Zeng J, Yue X (2011) Artificial plant optimization algorithm for constrained optimization problems. In: *2011 Second international IEEE conference on innovations in bio-inspired computing and applications*. Shenzhen, China, pp. 120–123. https://doi.org/10.1109/IBICA.2011.34
42. Ghaemi M, Feizi-Derakhshi MR (2014) Forest optimization algorithm. Expert Syst Appl 41(15):6676–6687. https://doi.org/10.1016/j.eswa.2014.05.009
43. Cheraghalipour A, Hajiaghaei-Keshteli M, Paydar MM (2018) Tree growth algorithm (TGA): a novel approach for solving optimization problems. Eng Appl Artif Intell 72:393–414. https://doi.org/10.1016/j.engappai.2018.04.021
44. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680. https://doi.org/10.1126/science.220.4598.671
45. Irizarry R (2005) A generalized framework for solving dynamic optimization problems using the artificial chemical process paradigm: applications to particulate processes and discrete dynamic systems. Chem Eng Sci 60(21):5663–5681. https://doi.org/10.1016/j.ces.2005.05.028
46. Melin P, Astudillo L, Castillo O, Valdez F, Garcia M (2013) Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm. Expert Syst Appl 40(8):3185–3195. https://doi.org/10.1016/j.eswa.2012.12.032

47. Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 110:69–84. https://doi.org/10.1016/j.advengsoft.2017.03.014
48. Birbil SI, Fang SC (2003) An electromagnetism-like mechanism for global optimization. J Global Optim 25(3):263–282. https://doi.org/10.1023/A:1022452626305
49. Xie L, Zeng J, Cui Z (2009) General framework of artificial physics optimization algorithm. In: *2009 IEEE world congress on nature and biologically inspired computing (NaBIC)*. Coimbatore, India, pp 1321–1326. https://doi.org/10.1109/NABIC.2009.5393736
50. Abedinpourshotorban H, Shamsuddin SM, Beheshti Z, Jawawi DN (2016) Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm. Swarm Evol Comput 26:8–22. https://doi.org/10.1016/j.swevo.2015.07.002
51. Yadav A (2019) AEFA: artificial electric field algorithm for global optimization. Swarm Evol Comput 48:93–108. https://doi.org/10.1016/j.swevo.2019.03.013
52. Kaboli SHA, Selvaraj J, Rahim NA (2017) Rain-fall optimization algorithm: a population based algorithm for solving constrained optimization problems. J Comput Sci 19:31–42. https://doi.org/10.1016/j.jocs.2016.12.010
53. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184. https://doi.org/10.1016/j.ins.2012.08.023
54. Shah-Hosseini H (2011) Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int J Comput Sci Eng 6(1–2):132–140. https://doi.org/10.1504/IJCSE.2011.041221
55. Taillard ED, Voss S (2002) POPMUSIC—Partial optimization metaheuristic under special intensification conditions. Essays and surveys in metaheuristics operations research/computer science interfaces series. Springer, Boston, MA, pp 613–629. https://doi.org/10.1007/978-1-4615-1507-4_27
56. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194(36–38):3902–3933. https://doi.org/10.1016/j.cma.2004.09.007
57. Purnomo HD (2014) Soccer game optimization: fundamental concept. Jurnal Sistem Komputer 4(1):25–36
58. Osaba E, Diaz F, Onieva E (2014) Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. Appl Intell 41(1):145–166. https://doi.org/10.1007/s10489-013-0512-y
59. Razmjooy N, Khalilpour M, Ramezani M (2016) A new meta-heuristic optimization algorithm inspired by FIFA world cup competitions: theory and its application in PID designing for AVR system. J Control Auto Electr Sys 27(4):419–440. https://doi.org/10.1007/s40313-016-0242-6
60. Juarez JRC, Wang HJ, Lai YC, Liang YC (2009) Virus optimization algorithm (VOA): A novel metaheuristic for solving continuous optimization problems. In: *2009 Asia pacific industrial engineering and management systems conference (APIEMS 2009)*, pp 2166–2174.
61. Li MD, Zhao H, Weng XW, Han T (2016) A novel nature-inspired algorithm for optimization: virus colony search. Adv Eng Softw 92:65–88. https://doi.org/10.1016/j.advengsoft.2015.11.004
62. Chen TC, Tsai PW, Chu SC, Pan JS (2007) A novel optimization approach: bacterial-GA foraging. In: *Second international IEEE conference on innovative computing, information and control (ICICIC 2007)*. Kumamoto, Japan, pp 391. https://doi.org/10.1109/ICICIC.2007.67
63. Chu Y, Mi H, Liao H, Ji Z, Wu QH (2008) A fast bacterial swarming algorithm for high-dimensional function optimization. In: *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)*. Hong Kong, China, pp 3135–3140. https://doi.org/10.1109/CEC.2008.4631222
64. Das S, Biswas A, Dasgupta S, Abraham A (2009) Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. In: Abraham A, Hassanien AE, Siarry P, Engelbrecht A (eds) Foundations of computational intelligence studies in computational intelligence, vol 3. Springer, Berlin, Heidelberg, pp 23–55. https://doi.org/10.1007/978-3-642-01085-9_2
65. Niu B, Wang H (2012) Bacterial colony optimization: principles and foundations. In: Huang DS, Gupta P, Zhang X, Premaratne P (eds) Emerging intelligent computing technology and applications (ICIC 2012), communications in computer and information science, vol 304. Springer, Berlin, Heidelberg, pp 501–506
66. Uymaz SA, Tezel G, Yel E (2015) Artificial algae algorithm (AAA) for nonlinear global optimization. Appl Soft Comput 31:153–171. https://doi.org/10.1016/j.asoc.2015.03.003
67. Yang XS (2020) Nature-inspired optimization algorithms: challenges and open problems. J Comput Sci 46:101104. https://doi.org/10.1016/j.jocs.2020.101104

68. Dhal KG, Das A, Ray S, Galvez J, Das S (2020) Nature-inspired optimization algorithms and their application in multi-thresholding image segmentation. Arch Comput Methods Eng 27(3):855–888. https://doi.org/10.1007/s11831-019-09334-y
69. Tzanetos A, Dounias G (2020) A comprehensive survey on the applications of swarm intelligence and bio-inspired evolutionary strategies. In: Tsihrintzis G, Jain L (eds.) Machine learning paradigms. Learning and analytics in intelligent systems (vol 18). Springer, Cham, pp.337–378. https://doi.org/10.1007/978-3-030-49724-8_15
70. Petrosillo N, Viceconte G, Ergonul O, Ippolito G, Petersen E (2020) COVID-19, SARS and MERS: Are they closely related? Clin Microbiol Infect 26(6):729–734. https://doi.org/10.1016/j.cmi.2020.03.026
71. Hethcote HW (2000) The mathematics of infectious diseases. SIAM Rev 42(4):599–653. https://doi.org/10.1137/S0036144500371907
72. Kyagulanyi A, Muhanguzi JT, Dembe O, Kirabo S (2020) Risk analysis and prediction for COVID-19 demographics in low resource settings using a python desktop app and excel models. MedRxiv. https://doi.org/10.1101/2020.04.13.20063453
73. Sy KTL, White LF, Nichols BE (2020) Population density and basic reproductive number of COVID-19 across United States counties. MedRxiv. https://doi.org/10.1101/2020.06.12.20130021
74. Liu Y, Gayle AA, Wilder-Smith A, Rocklov J (2020) The reproductive number of COVID-19 is higher compared to SARS coronavirus. J Travel Med. https://doi.org/10.1093/jtm/taaa021
75. Biswas K, Khaleque A, Sen P (2020) Covid-19 spread: reproduction of data and prediction using a SIR model on Euclidean network. ArXiv preprint
76. Zhang Y, You C, Cai Z, Sun J, Hu W, Zhou XH (2020) Prediction of the COVID-19 outbreak based on a realistic stochastic model. MedRxiv. https://doi.org/10.1101/2020.03.10.20033803
77. Liu Z, Magal P, Seydi O, Webb G (2020) A COVID-19 epidemic model with latency period. Infect Dis Model 5:323–337. https://doi.org/10.1016/j.idm.2020.03.003
78. Kermack WO (1927) McKendrick AG (1991) Contributions to the mathematical theory of epidemics–I. Bull Math Biol 53(1–2):33–55. https://doi.org/10.1007/bf02464423
79. da Silveira MP, da Silva Fagundes KK, Bizuti MR, Starck E, Rossi RC, e Silva DTDR (2020) Physical exercise as a tool to help the immune system against COVID-19: an integrative review of the current literature. Clin Exp Med 21:15–28. https://doi.org/10.1007/s10238-020-00650-3
80. Taghizadeh-Hesary F, Akbari H (2020) The powerful immune system against powerful COVID-19: a hypothesis. Med Hypotheses 140:109762. https://doi.org/10.1016/j.mehy.2020.109762
81. An P, Chen H, Ren H, Su J, Ji M, Kang J, Jiang X, Yang Y, Li J, Lv X, Yin A, Chen D, Chen M, Zhou Z, Dong W, Ding Y, Yu H (2020) Gastrointestinal symptoms onset in COVID-19 patients in Wuhan China. Dig Dis Sci. https://doi.org/10.1007/s10620-020-06693-6
82. Craw S (2011) Manhattan distance. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning. Springer, Boston, MA, p 639
83. Abdullahi M, Ngadi MA (2016) Symbiotic organism search optimization based task scheduling in cloud computing environment. Futur Gener Comput Syst 56:640–650. https://doi.org/10.1016/j.future.2015.08.006
84. Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34(3):209–219. https://doi.org/10.1016/j.omega.2004.10.004
85. Reinelt G (1991) ATT48 from TSPLIB—A traveling salesman problem library. https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.htmlAccessed 26 July 2020.