Applied Soft Computing 43 (2016) 596–618

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Virulence Optimization Algorithm

Morteza Jaderyan, Hassan Khotanlou *

Department of Computer Engineering, Bu-Ali Sina University, Iran

ARTICLE INFO

Article history:
Received 6 November 2014
Received in revised form
25 November 2015
Accepted 25 February 2016
Available online 2 March 2016

Keywords:
Optimization
Virulence
Virus cloning
Host environment
Continuous and non-linear functions

ABSTRACT

In this paper, a new optimization algorithm to solve continuous and non-linear optimization problems is introduced. This algorithm is inspired by the optimal mechanism of viruses when infecting body cells. Special mechanism and function of viruses which includes the recognition of fittest viruses to infect body cells, reproduction (cloning) of these cells to prompt "invasion" operation of ready-to-infect regions and then escaping from infected regions (to avoid immune reaction) is the basis of this evolutionary optimization algorithm. Like many evolutionary algorithms, the Virulence Optimization Algorithm (VOA) starts the optimization process with an initial population consisting of viruses and host cells. The host cell population represents the resources available in host environment or the region containing the global optimum solution. The virus population infiltrates the host environment and attempts to infect it. In the optimization procedure, at first the viruses reside in the constituted regions or clusters of the environment called virus groups (via K-means clustering). Then they scatter in host environment through mutation (Drifting) and recombination (Shifting) operators. Then among the virus groups, the group with highest mean fitness is chosen as escape destination. Before the escape operation commences, the best viruses in each virus group are recognized and undergoes a cloning operation to spread the Virulence in the host environment. This procedure continues until the majority of the virus population is gathered in the region containing the maximum resources or the global optimum solution. The novelty of the proposed algorithm is achieved by simulating three important and major mechanisms in the virus life, namely (1) the reproduction and mutation mechanism, (2) the cloning mechanism to generate the best viruses for rapid and excessive infection of the host environment and (3) the mechanism of escaping from the infected region. Simulating the first mechanism in the virus life enables the proposed algorithm to generate new and fittest virus varieties. The cloning mechanism facilitates the excessive spread of the fittest viruses in the host environment to infect the host environment more quickly. Also, to avoid the immune response, the fittest viruses (with a great chance of survival) are duplicated through the cloning process, and scattered according to the Vicinity Region Radius of each region. Then, the fittest viruses escape the infected region to reside in a region which possess the resources necessary to survive (global optimum). The evaluation of this algorithm on 11 benchmark test functions has proven its capability to deal with complex and difficult optimization problems.

1. Introduction

In computer science, mathematical optimization is referred to a process in which the best option among a set of candidate options is chosen according to a specific criteria. In its simplest form, an optimization problem consists of minimizing or maximizing a real-valued function. In this process, input values are chosen from a set of allowed values for input values systematically and then the value of real-valued function is calculated [1].

In other words, the optimization process can be regarded as improving a concept or an object in which the inputs or characteristics of a device, mathematical operation or a specific experiment is well-adjusted so the outputs or the results is minimized or maximized. The input of this process consists of variables, a process or function which is recognized as a cost (fitness) function and an output which displays the cost or fitness [2,3].

Algorithms and methods of optimization are categorized under five different categories:

1. Combinational optimization. It concerns with finding the optimal solution (object) among a finite set of solutions (objects).

* Corresponding author at: Bu-Ali Sina University, Fahmideh Ave., Hamedan, Iran. Tel.: +98 813 827 2072.
E-mail addresses: m.jaderyan92@basu.ac.ir (M. Jaderyan), khotanlou@basu.ac.ir (H. Khotanlou).

http://dx.doi.org/10.1016/j.asoc.2016.02.038
1568-4946/

Exhaustive search will be avoided through a selection of finite set of feasible solutions or conversion of solution set to a discrete and finite solution set. Applications of this type of optimization methods include development of optimal airline networks, optimal postal delivery route [4,5].
2. Dynamic Programming (DP). It is the process of breaking a complicated problem into a number of simpler sub-problems. In general, to solve a given problem using DP, all the sub-problems are solved. Then the solutions for these sub-problems are combined to obtain the final optimal solution. The DP method is one of the most popular optimization methods. This method is applicable to problems with overlapping sub-problem and optimal sub-structure feature. The DP method is also considered an "intelligence Brute-Force" method for finding optimal solutions [6,7].
3. Gradient method. Very efficient type of optimization algorithm to solve a given problem in the form of "$min(f(x))$". One of the most important gradient methods is "Gradient decent" which is a first-order optimization algorithm. To find the local minima or maxima of a given function, gradient descent method employs the negative or positive value of gradient at a certain point [8].
4. Stochastic optimization. These methods are based on generating value for variables which appear in optimization equation and then using these generated values to solve a specified problem. Stochastic methods to solve optimization problems also comprise "random iteration" methods and a combination of "random Iteration" and "random objective function" methods [9,10].
5. Evolutionary algorithms. Evolutionary algorithms are heuristic methods that employ organic evolution-inspired techniques such as mutation, recombination and natural selection to find the optimal state of a specific system under a specific set of constraints.

The main idea and contribution of this paper can be described as follows: "introducing a novel evolutionary optimization algorithm inspired by the virus life mechanisms of infecting host environment and finding the most suitable residence to guarantee the growth and survival of the fittest viruses". The main objective of this paper is to determine the performance and the efficiency of the proposed algorithm in solving the standard optimization problems. The performance and efficiency of the proposed algorithm are evaluated in terms of convergence speed and the required operational time to reach the optimal solution. The following mechanisms and processes are simulated in the proposed algorithm:

1. Variation mechanisms. One of the most vital components of the proposed algorithm is the variation mechanism. Variation leads to the creation of new virus varieties and facilitates the virus adaptation to the ever-changing environment. To simulate this phenomenon, the viruses' properties are combined or undergo mutation to create new virus types. Also, the implemented mutation mechanism employs a varying mutation step-size. The fittest viruses have a greater chance of survival and participate in spreading of virulence (Section 3.2).
2. The cloning process. The excessive spread of viruses in host environment is achieved by the rapid cloning of viruses. This characteristic enables the viruses to search the host environment (candidate solution space) more thoroughly. The cloned viruses are scattered according to Vicinity Region Radius of each region (see Section 3.3). Also, the main requirement for the escape mechanism (see Section 3.4) is the cloning process.
3. Escape mechanism. The viruses are always in search of location in candidate solution space which possess the maximum resources (global optimal solution). After rapid reproduction and cloning of the viruses, in order to avoid the immune response and search for a better suited living environment, the fittest viruses escape to a region containing the vital resources necessary to survive.
4. The process of maintaining equilibrium. This process is done by eliminating the worst fitted viruses (see Section 3.5).

1.1. Related works

Nature-inspired optimization method usually starts off with an initial set of variables. Then they get on the path of evolution to obtain a global minimum or maximum of the objective function. Genetic algorithm (GA) is one of the most popular techniques in the researches related to evolutionary algorithms. The genetic algorithm employs a variety of evolutionary operators which directly or indirectly are inspired by natural variation and selection [2,11,12]. At first, the GA generates a set of candidate solutions from the initial population. Generating a random population of candidate solutions allows the exploration of the entire search space to find the optimal solutions. The size of the initial population depends on the underlying optimization problem. Finally, during consecutive iterations of this algorithm and by using operators such as mutation, recombination and selection, the initial population will converge to a region that contains the global optimum solution.

One of the most prominent features of the evolutionary algorithms is their ability to reproduce the optimal population and search the candidate solution space with the aim of finding the global optimum solution. From this perspective, we can categorize the evolutionary algorithm into six categories [44]:

1. Differential Evolution-based Approaches. The differential evolution (DE) algorithm [45,46] was introduced in 1995 by storn and price. This algorithm uses weighted difference between two candidate solution to cause variation in the population and to produce viable candidate solutions. The newly formed candidate solutions inherit their feature partly from candidate solutions and partly from the old population. The Differential Evolution-based algorithm was initially proposed and designed to tackle the scalar optimization. However, due to its simplicity to implement and efficiency, it can be used to solve the multi-objective optimization problems. It is worth noting that the Differential Evolution algorithm has been extended to solve discrete or mixed discrete and continuous optimization problems.Algorithms such as Pareto-frontier differential evolution (PDE) which was proposed in [47] and the Differential Evolution based on Pareto-adaptive ε-dominance and orthogonal design which was introduced in [48] are among the algorithms that were inspired by this type of evolutionary computing approach.
2. Immune-based approaches. This type of evolutionary algorithm is inspired by the defence mechanisms of the natural immune system against pathogens. Because of the clonal selection and the affinity maturation caused by hyper-mutation, the immune system is capable of adapting immune cells (B-cells) to new varieties of antigens. By simulating this phenomenon, the artificial immune system algorithms were invented to tackle and solve optimization problems.As mentioned earlier, artificial immune system (AIS) is a class of intelligent computing systems which is inspired by the principles and processes of the human body's immune system. AIS algorithm uses the learning feature and memory capability of natural immune system agents to solve optimization problems. AIS algorithms are adaptive systems inspired by theoretical immunology and observed immune functions, principles and models [17,18] which use the accumulated intelligence of their agents to find the optimal solution in the search space. The unique features such as self-organizing ability, distributing ability, multi-layering and the memory associated with its agent has distinguished AIS from other similar

methods in machine learning and optimization. The performance of AIS relies on proper choice of mutation operator, negative selection and clonal selection. In [49], a non-dominated neighbour immune system is proposed. This algorithm emphasizes embeds a selection strategy which emphasizes more on less-crowded solution. Algorithms such as hybrid immune multi-objective optimization algorithm [50] and immune system based on a multiple-affinity model [51] are proposed based on immune system principles and mechanisms.

3. Particle Swarm-based Approaches. In this category, two important issues should be considered: (1) how to select the best local and global particles (which are considered leaders) to guide search of a particle representing optimum solution and (2) how to maintain the best solutions found so far in order to obtain the optimum solution. To overcome the issue of maintaining good solutions, a secondary population is usually employed to maintain the non-dominated solutions in search space [44]. The Particle Swarm-based algorithms can also be extended to multi-objective optimization. Moore and Chapman extended this idea to multi-objective optimization [52].The particle swarm optimization (PSO) is developed by Kennedy and Eberhart in 1995. This stochastic optimization algorithm is inspired by social behaviour of the birds' population or fish movement patterns [12–14]. In PSO every particle searches for an optimal solution. Particles are always moving, so each particle has a specific velocity. Finally, through cooperation and sharing and exchange of the information regarding the explored regions of the search space, the particles will find the optimal solution.Algorithms such as EA–PSO hybrid [53], fuzzy clustering-based PSO [54] and multi-objective comprehensive learning particle swarm optimizer (MOCLPSO) [55] are among the optimization algorithms which are based on the idea of particle swarming.
4. Probabilistic Model-based Approaches. Probabilistic Model-based evolutionary algorithms are a new computing paradigm, in the field of evolutionary computing. The most important characteristic of Probabilistic Model-based Approaches is that they do not employ the conventional crossover and mutation operators to create new candidate solutions. Instead, these algorithms explicitly extract the global statistical information from a previous search. Then, according to the extracted information, a probability distribution model is constructed to produce promising candidate solution. For this purpose, based on the extracted information, new candidate solutions are sampled from the constructed probabilistic model. Compared to traditional EA methods, probabilistic model-based methods emphasize the information about the population distribution rather than the information about the individual location [44]. The main issues with these methods include selection of the proper probabilistic model before the exaction of the algorithm, constructing the probability distribution model and consequently, sampling new candidate solutions from the model. The Ant Colony Optimization algorithm is an example of a probabilistic model-based evolutionary algorithm.Ant Colony Optimization (ACO) algorithm is an evolutionary algorithm which is inspired by the actual behaviour of ants in leaving a pheromone trail [12,15,16]. The basis of this method is the employment of intelligence obtained by the accumulation of agents to solve optimization problems. The first version of this method which takes inspiration from the behaviour of ants in finding the best path between food resources and living environment, attempts to find the optimal path in a graph. Over the past few years, various instances of this algorithm are developed to solve numerical optimization problems.

   The cross entropy (CE) method [56], the quantum-inspired genetic algorithm (QGA) [57,58] and the estimation of distribution algorithm (EDA) [59] are among the optimization algorithms that share the ideas and the principles of Probabilistic Model-based Approaches.
5. Simulated annealing-based approaches. Simulated annealing (SA) is a single-point-based global optimization technique which is inspired by annealing in metallurgy [44,60]. Due to its simplicity, the simulated annealing optimization algorithm is widely used to tackle scalar optimization problems. The key difference between simulated annealing and other optimization algorithms is in the mechanism of updating a candidate solution when the individual (offspring) is dominated by the properties of the parent. In such case, a special updating mechanism called simulated annealing updating rule is employed.In [61], when an individual (offspring) is dominated by the parent individual, the simulated annealing updating rule id used to select the next viable individual (offspring) instead of the first one. Also, other simulated annealing-based approaches [62] assign a probability of selection to each individual (offspring) according to a domination-based energy function to determine the best candidate solutions in the population.
6. Meta-heuristic-based approaches. The meta-heuristic algorithm is a higher level procedure, algorithm or heuristic designed with the aim of finding, creating or selecting a search algorithm to guide the process of finding and selecting an adequate solution to a desired optimization problem [63]. Since, all the evolutionary optimization algorithms search for and select the best candidate solutions from population, these approaches are also called population search-based techniques. One approach to classify meta-heuristic algorithms is single solution vs population-based searches. In single solution approaches, the focus is about modifying and improving a single candidate solution. While in a population-based search the purpose is to maintain and improve multiple candidate solutions and in most cases the characteristics of the population are used to navigate the search. Another classification is global vs. local search.The cuckoo optimization algorithm (COA) is an evolutionary algorithm which is inspired by the life of a bird family, called Cuckoo. Special lifestyle of these birds and their characteristics in egg laying and breeding has been the basic motivation for development of this evolutionary optimization algorithm. The performance of ICA relies on proper choice of operators such as "immigration" and the mechanism of elimination of cuckoos in worst habitats of search space [2].

   Imperialist Competitive Algorithm (ICA) is a computational method for solving various kinds of optimization problems. Like many evolutionary algorithms, ICA does not require the calculation of the gradient of the objective function in the optimization process. From specific point of view, the ICA is considered the social counterpart of the GA algorithm because ICA is a mathematical model and computer simulation of the social evolution of mankind while the GA is based on genetic evolution of other species. The performance of ICA relies on proper choice of operators such as "Assimilation", "Revolution" and the mechanism of elimination and merger of imperialist empires in the search space [19].

   Other meta-heuristics-based optimization algorithms includes tabu search [64,65], scatter search [66] and the GRASP approach [67].

Table 1 illustrates the different categories of evolutionary optimization approaches, their subsidiaries and properties.

Some of the related works in the field of evolutionary computing are listed as follows:

Ref. [23] proposes a new particle swarm algorithm based on a clonal selection algorithm which prevents premature convergence of the algorithm and guarantees the diversity of the population.

In [67], a novel optimization method inspired by a paradigm from nature is introduced. Chemical reactions are used as a

Table 1
Different categories of evolutionary optimization approaches, their subsidiaries and properties.

#	Algorithm	Category	Multi-objective capable	Subsidiary algorithms	Characteristics
1	Particle Swarm Optimization	Particle swarm-based	Yes		Guided search, global and local particles
2	Artificial immune system	Immune-based	Yes	Non-dominated neighbour AIS, hybrid immune multi-objective	Global search, population-based
3	Imperialist Competitive Algorithm	Meta-heuristic	Yes	–	Population-based, guided search
4	Differential Evolutionary Algorithm	Differential Evolutionary-based	Yes	Pareto-frontier differential evolution (PDE), Pareto-adaptive ε-dominance and orthogonal design	Weighted difference of two solutions
5	Ant Colony Optimization	Probability-based Models	Yes	EA–PSO hybrid, fuzzy clustering-based PSO, MOCLPSO	Intelligence accumulation, probability distribution model
6	Cuckoo optimization algorithm	Meta-heuristic	Yes	–	Global search, population-based
7	Simulated Annealing optimization	Simulated Annealing-based	No (yes, if incorporated with other frameworks)	–	Single-point-based, updating rule
8	Virulence Optimization Algorithm (Proposed)	Meta-heuristic	Yes	–	Global search, population-based, guided search

paradigm to propose an algorithm that can be considered as a general purpose optimization technique. The proposed chemical reaction algorithm is a meta-heuristic strategy that performs a stochastic search for optimal solutions within a defined search space. In this optimization strategy, every solution is represented as an element (or compound), and the fitness or performance of the element is evaluated in accordance with the objective function. The 4 chemical reactions considered in this approach are the synthesis, decomposition, single and double substitution reactions. The objective of these operators is exploring or exploiting new possible solutions within a slightly larger hypercube than the original elements/compounds, but within the previously specified range. The synthesis and decomposition reactions are used to diversify the resulting solutions; these procedures showed to be highly effective to rapidly lead the results to a desired value.

In [68], an artificial immune system-based many-objective optimization algorithm with network activation scheme is introduced. This paper extends an AIS-based optimization algorithm to solve such many-objective optimization problems. The idea of ε-dominance and the holistic model of the immune network theory have been adopted to enhance the exploitation ability aiming for a quick convergence. In general, an AIS-based multi-objective optimization algorithm first generates the initial population. This population will go through cloning, variation, evaluation, network suppression and memory updating. Cloning and variation process generates modified solutions. Evaluation process assesses the fitness of each solution. A different fitness assignment scheme was proposed to increase the chance of domination. The ε-dominance is one of them which relaxes the dominance requirement. The resulting solutions are often dominated by solutions with lower or equal fitness in all objectives which are close enough to non-dominated solutions. Memory updating process selects a population for the next generation. The results show the potential of the AIS-based many-objective optimization algorithm.

1.2. Evolutionary applications

Nowadays optimization algorithms are vastly applied to solve optimization problems in areas of interest such as industrial planning, resource allocation, scheduling, decision making, machine learning and pattern recognition [20,21]. Moreover, optimization techniques are widely applied in branches of science, such as chemistry, commerce, engineering and computer science.

Ref. [22] introduces a new improved pattern recognition method based on an artificial immune system algorithm which is applied to improve the recognition capability of the cancer mass detection system. The improved AIS system alongside the K-nearest neighbour (KNN) algorithm constitutes a new method of recognition called AIS-KNN. This new improved method of pattern recognition is then employed to detect a cancer mass in medical images.

In [69], a new approach for dynamic fuzzy logic parameter tuning in Ant Colony Optimization and its application in fuzzy control of a mobile robot is introduced. The present paper explores a new fuzzy approach for diversity control in Ant Colony Optimization. The main idea is to avoid or slow down full convergence through the dynamic variation of a particular parameter. The formulated strategy is outperformed by AS and EAS in the membership function optimization problem, but managed to outperform other standard methods.

Table 2 summarizes the major applications of evolutionary algorithms [44].

Main features of evolutionary algorithms which distinguish them from other similar optimization techniques are [2,12]:

(1) Being robust to dynamic changes: Traditional methods of optimization are not robust to dynamic changes in the environment and they require a complete restart for providing a solution. In contrary, evolutionary computation can be used to adapt solutions to changing circumstances.
(2) Broad applicability: Evolutionary algorithms can be applied to any problems that can be formulated as function optimization problems.
(3) Hybridization with other methods: Evolutionary algorithms can be combined with more traditional optimization techniques.
(4) Solves problems that have no solutions: one of the main advantages of evolutionary algorithms is that they can be applied to domains which no suitable human expertise is available.

Considering these features, evolutionary algorithms can be applied to various applications such as control and power system operation [21], NP-hard combinational problems, chemical processes [20], mobile communication networks, navigation and image processing.

Table 2
Summary of applications of EAs in real world problems.

#	Area of interest	Details of application	Types of evolutionary algorithm used
1	Scheduling heuristics	Planning	A multi-objective particle swarm (MOPS) [70]
2	Data mining and rule extraction	Data mining	A Pareto-based DE algorithm [71].
		Rule extraction	A genetic cooperative competitive learning (GCCL) [72]
3	Assignment and management	Resource allocation	Tabu search with the combination of a dominance rule [64]
4	Bioinformatics	DNA sequence design	A constrained controlled elitist NSGA-II [73]
5	Control systems and robotics	Control scheme design	A prioritized multi-objective stochastic algorithm based on SA (PMOSA) [74]
6	Pattern recognition and image processing	Image processing	A bi-objective EA [75]
		Pattern classification	An EA for the unsupervised learning and data clustering Problems [76]
7	Others	Fault diagnosis	An artificial immune algorithm [77]
		Website	A grammar-based genetic programming algorithm [78]

The evolutionary history of viruses and their mechanisms of infecting and spreading the virulence in the host environment has exhibited the extraordinary ability of this parasitic variety to adapt themselves to the ever-changing host environment. Thereupon, the mature viruses are fully capable of finding the optimal residence to guarantee their survival and growth. The main driving forces behind such a process are the variation operators such as mutation and reproduction and cloning mechanism. Simulating the adapting nature of virus life, the infiltration of the host environment, virus reproduction, mutation and cloning and eventually spreading the virulence in the host environment have been the basic motivation for development of this new evolutionary algorithm. Such processes lead to selection of the most suitable environment for the survival of the viruses and simulating them plays a very important role in guaranteeing adequate performance of the proposed virulence optimization algorithm. Also, another objective of this research is the evaluation of performance and the convergence speed of the proposed algorithm in solving optimization problems and generating optimal solutions.

In this paper, we introduce a new optimization algorithm to solve optimization problems which is inspired by the optimal mechanism of viruses when infecting body cells. Special mechanism and function of viruses which includes the recognition of fittest viruses to infect body cells, reproduction (cloning) of these cells to prompt "invasion" operation of ready-to-infect regions and then escaping from infected regions (to avoid immune reaction) is the basis of this evolutionary optimization algorithm. The initial population in this algorithm comprises viruses and body cells. In preliminary generations, the body cell population is greater than virus population, but over time and during the evolution process (with mass reproduction of viruses) the number of viruses exceeds the body cells population size and environment converges to a population with more virus and fewer body cell. In other words, during the evolution process more virus entity will be produced and in turn these produced viruses will participate in the infecting process of body cells. Finally, after a sufficient number of iterations, the virus population in environment converges to a specific region of solution space which possesses the most resources (fittest solutions). This optimal region contains the global optimum solution of a given problem. This proposed algorithm employs operators such as reproduction, mutation, reproduction (clone) and escape to reach a global optimum solution. This paper illustrates how the operation of infecting body cells by viruses is modelled and implemented.

Second section of this paper investigates the virus life and its variation mechanisms. In this section, various theories regarding virus evolution, the concept of "virulence" and variation operators, as the driving force behind virus evolution is investigated. In the third section, the proposed method and its embedded mechanisms to reach the global optimum of a given problem is introduced and studied in detail. In section four, the proposed method is evaluated using standard test functions and the results are displayed and compared to other well-known methods of optimization. In Section 5, the discussion about the results of the experiments is presented. Finally the conclusions are presented in Section 6.

2. Viruses and their reproduction towards virulence

A subfield of evolutionary biology, which is specifically concerned with evolution of viruses is called viral evolution [24]. Most known viruses have very short life span and very high mutation rate (mutation per gene). The high mutation rate in combination with natural selection allows viruses to quickly adapt themselves to rapid changes in the host environment.

2.1. Evolution of viruses

The evolutionary history of viruses can to some extent be inferred from analysis of contemporary viral genomes. The mutation rates for many viruses have been measured, and application of a molecular clock allows dates of divergence to be inferred [24,25]. Evolution of viruses is possible through the changes in DNA or RNA of viruses. Through these fundamental changes, the fittest mutated viruses which can adapt themselves to changing circumstances of environment the best possible manner, rapidly replace the infected body cell and they rapidly grow in the host environment. This type of rapid reproduction of viruses is according to Darwinian Theory of evolution. The manner in which the viruses reproduce and grow in the host environment, makes them prone to genetic changes which is the driving force behind their evolution.

In host cells, there are mechanisms for correcting mistakes when DNA replicates and these kicks in whenever cells divide [24,26]. These important mechanisms prevent potentially lethal mutations from being passed on to offspring. But these mechanisms do not work for RNA and when an RNA virus replicates in its host cell, changes in their genes are occasionally introduced in error, some of which are lethal. One virus particle can produce millions of progeny viruses in just one cycle of replication, therefore the production of a few "dud" viruses is not a problem. Most mutations are "silent" and do not result in any obvious changes to the progeny viruses, but others confer advantages that increase the fitness of the viruses in the environment. These could be changes to the virus particles that disguise them so they are not identified by the cells of the immune system or changes that make antiviral drugs less effective [24,27].

When two viruses with similar genetic strains try to infect a particular host cell, they can shuffle their genes. This phenomenon called Genetic Shifting and usually results in the creation of a new type of virus. Other viruses evolve when their genetic changes and the mutations done to their genes accumulate over time. This phenomenon called Genetic Drift [24,28].

But the main question that may come to mind is how the viruses evolve rapidly? To answer this question it should be mentioned that evolution is generally based on change and variation. Unique biological features (mentioned above) allow them to rapidly produce different types of new viruses. On the other hand, viruses have a high mutation rate, produce a large number of offspring and have

very short life span. Moreover, other biological features such as Reassortment allow viruses to produce different types of new viruses [29,30].

But why different types of viruses, cause different type of harm to their host cell? To answer this question we should introduce the concepts of "Virulence". The amount of harm and damage that viruses impose to their host cell is called "Virulence". The manner in which the viruses transmit to their host environment has a huge influence on virulence evolution of viruses. Viruses that mostly spread through the transmission to off-spring host cells (Vertical Transmission) have lower virulence evolution rate than viruses with Horizontal Transmission. On the other hand, the more quickly the viruses transmit and spread, the higher virulence evolution rate they have [29,30]. So, for better understanding of viruses' mechanism of infecting host cells, we first explain the concept of Virulence and its function.

2.2. Virulence

Many of recent theories regarding viral evolution are based on the hypothesis that there is a direct relation between virulence and virus transmission. The theory which describes this relation is called "Virulence – Transmission Trade-off Hypothesis". The trade-off hypothesis states that virulence is an unavoidable consequence of virus transmission. It is not possible for the virus to increase the duration of an infection without paying a cost. In this case a particular trade-off exists between diffusion (spread) levels of virulence and the duration of an infection. In other words, a virus can choose either a short life span with high virulence level or a longer life span with lower virulence level. In the former scenario, virus causes more harm and infection of its host cell, but very quickly becomes the subject of immune response. In the latter scenario, the virus can adapt itself to the changing circumstances of environment very well and learns to avoid the immune response of natural immune system, but this covert operation results in a very low virulence rate in the host environment. The fitness of viruses in host environment is specified with "baseline reproduction rate" equation [29]. This equation states that the fitness of viruses is equal to the product between the numbers of infections caused by a single infected host per unit of time multiplied by the duration of the infection. This trade-off is illustrated by a curve also known as "Trade-off Curve" (Fig. 1).

Parameters μ, α^*, β^* and γ^* are the parameters involved in calculation of baseline reproduction rate [29]. If the shape of this curve is linear or convex, system converges to a short-term infection with infinite transmission rate.

Natural selection and survival selection play important roles in the evolution of viruses. There will be a selection in favour of those varieties of viruses which vegetate in host environment whence they can escape, so they do not become subject of immune response. The surviving varieties of viruses adapt themselves more particularly to the conditions surrounding the invasion and escape. So, in survival selection stage the viruses which perform better in invasion and escape phase have a higher chance of selection than the other viruses in the host environment. But the surviving varieties of viruses would gradually lose their highly virulent invasive qualities. In other words, their mortality rate or their chance of survival in host environment decreases over time and this results in rapid detection by the immune system and consequent elimination due to immune response [29]. The short-sighted evolution theory describes this process the best [29,31]. This theory states that in a diverse population of virus varieties, the varieties which have a high transmission rate and therefore have higher fitness and virulence, have a greater chance of survival. This is due to the fierce competition of viruses to obtain vital but limited resources for survival. Assuming that there is a direct relation between virulence and transmission rate, the result of this kind of competition between viruses is that the host's immune system will become aware of foreign invaders and consequently viruses with worst fitness will be eliminated. The experiments indicate that viruses with highest virulence level not only win the competition to obtain resources but also have a higher transmission rate. However, it should be noted that if a virus with best fitness always wins the competition, the diversity of virus population decreases and this is not acceptable because this results in rapid destruction of resources available on the environment and consequently rapid detection and elimination of viruses [29,31] (the mortality rate of host cells quickly increases).

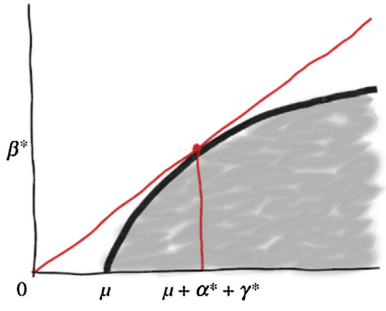

Fig. 1. Trade-off Curve depicting fitness of viruses determined by the baseline reproduction rate.

2.3. Operators of virulence

Evolution is based on variation and reproduction. Reproduction in virus varieties is the combination of replication (cloning) and genetic changes similar to mutation. Variation in virus varieties is the direct consequence of their mutation.

2.3.1. Reproduction

Reproduction is based on "self-enhance" property and results in a staggering increase in the number of viruses in the host's environment [31]. The main core of reproduction is "replication" or "cloning" operation. Replication is like a manual which contains instructions regarding reproduction and gets activated during the reproduction procedure. The more accurate the replication is, the more perfect and errorless inheritance from parent to off-spring occurs. However, sometimes imprecise inheritance occurs and results in reproduction error. This kind of imprecise process results in genetic changes similar to mutations in viruses. Therefore, in reproduction two offspring will be produced [29–31]:

(1) An off-spring which is quite similar to parent virus and inherits all the genetic features of the parent.
(2) A second off-spring which, as the result of a process similar to mutation, specific changes are made to its genes and is quite different from its parent.

However, it is possible for reproduction to produce only one offspring that bear very different genetic features compared to its parent [31]. The process of reproduction in virus varieties is illustrated in Fig. 2.

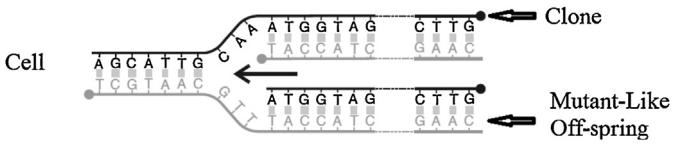

Fig. 2. Representation of virus reproduction.

Fig. 3. Representation of virus mutation.

2.3.2. Mutation (Drifting)

The genetic changes that occur in virus genome is called mutation. In other words, mutation is the irreversible changes that occur in RNA or DNA of viruses. Since the growth and reproduction rate of viruses is very high, then the mutation in their genome over a short period of time is very probable [32–35]. There are two major factors in virus mutation [29,31,35–37]:

(1) Mutation rate. The changes in this factor are very dependent on a virus generation time and its capability to adapt to the host's environment.
(2) Mutation frequency. Is the counterpoint of mutation rate and usually measures the genetic changes created in a population.

It should be noted that for an evolutionary system the most important factor is the mutation rate. Fig. 3 illustrates the mutation operation in viruses.

When a virus mutates and starts to infect the host cells, it causes severe changes in their genetic structure. The more the genetic changes in genomes are, the more diverse the population is.

2.3.3. Recombination (Shifting)

The high similarities between different varieties of viruses in different generations indicates the existence of recombination operation in their reproduction cycle. Recombination results either in elongation of genetic DNA strands of viruses or increase in the virulence level of viruses and infected host cells [29,31,32,38]. When two different viruses intend to infect the same host cell, they exchange their genetic materials. Two major types of recombination in viruses are [38]:

(1) Re-assortment of segments. The parent and their genetic structure get fragmented and then parent swap their gene fragments.
(2) Intra-molecule recombination. In this type of recombination, the genome of viruses is made of template-like structures which viruses swap.

An example of virus recombination is illustrated in Fig. 4.

3. The proposed Virulence Optimization Algorithm

Like many evolutionary algorithms, the Virulence Optimization Algorithm (VOA) starts the optimization process with an initial population consisting of viruses and host cells. Host cells represent regions of the solution space (feature space) of the problem which viruses intend to invade and infect. Viruses are always looking for regions of solution space which contain the maximum resources vital for survival. The viruses which have a maximum affinity (similarity) to the host cells have a greater chance of infecting host environment without triggering the immune response. After infecting host cells (through reproduction and genetic mutation of their genome), these viruses spread in host environment and infect more host cells. Therefore, we can express that the Virulence Optimization Algorithm in its initial stage has a population which comprises mostly host cells and a few virus cells that successfully entered the host environment without triggering an immune response. If during the process of optimization via VOA algorithm the viruses are gathering around a specific environment in the region, it means that this region has the maximum resources for reproduction, growth and ultimately survival of viruses.

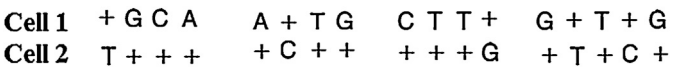

Fig. 4. Representation of virus reproduction.

So the region in an environment which possesses the maximum resources necessary to reproduce and survive will be the term that VOA intends to optimize. Fig. 5 illustrates the flow diagram of the Virulence optimization algorithm.

As mentioned earlier, viruses are always looking for regions of solution space which contain the maximum resources vital for survival. The region with maximum resource, increases the chance of virus for survival, reproduction and infecting host cells. As soon as they spread in their host environment, they begin to make clusters or regions in the search space. Each virus, in whichever cluster or region they are in, begins to search for region with maximum resources through mutation and recombination operation. Virus reproduction allows them to escape to region with maximum resources vital for survival as soon as they find it. The operation of escape to the region containing maximum resources serves two crucial purposes:

(1) Viruses will reside in regions with maximum resources. This results in much greater chance of survival, reproduction, growth and contaminating the environment for viruses and their consequent offspring.
(2) This escape decreases the probability of detection of fittest mature viruses by the body's immune system which will cause in triggering an immune response and their subsequent elimination.

The viruses which escape to better regions (containing maximum resources) are most likely the viruses with the highest chance of survival, reproduction and growth. In other words, these viruses have higher fitness compared to other viruses. The viruses with lower fitness will be detected by the immune system due to their excessive reproduction. This will result in activation of immune response and ultimately elimination of these viruses. The surviving variety of viruses begin to grow, mutate and reproduce in their new residing region. This procedure will resume until the vast majority of surviving viruses converges to a region which contains most vital resources. This state indicates that viruses are resided in the global optimum region with maximum resources for growth and reproduction.

3.1. Generating initial virus and cell population

In most cases to solve an optimization problem, the values of problem variables will be represented in the form of an array. For example, in GA this array is called "chromosome" and in cuckoo optimization algorithm (COA) this array is called "Habitat". In the proposed virulence optimization algorithm this array is called "gene strand". In an N_{var}-dimensional optimization problem, a gene strand is an array of $1 \times N_{var}$ which can be regarded as the geographical location of a virus or the host cell in problem's search space. This array can be defined as follows:

$$gene \quad Strand = \{g_1, \ldots, g_{N_{var}}\} \tag{1}$$

Each value of variables $\{g_1, \ldots, g_{N_{var}}\}$ is a real-valued umber, although it can be represented by integer numbers or string. In other word, the representation of variables of the problem can

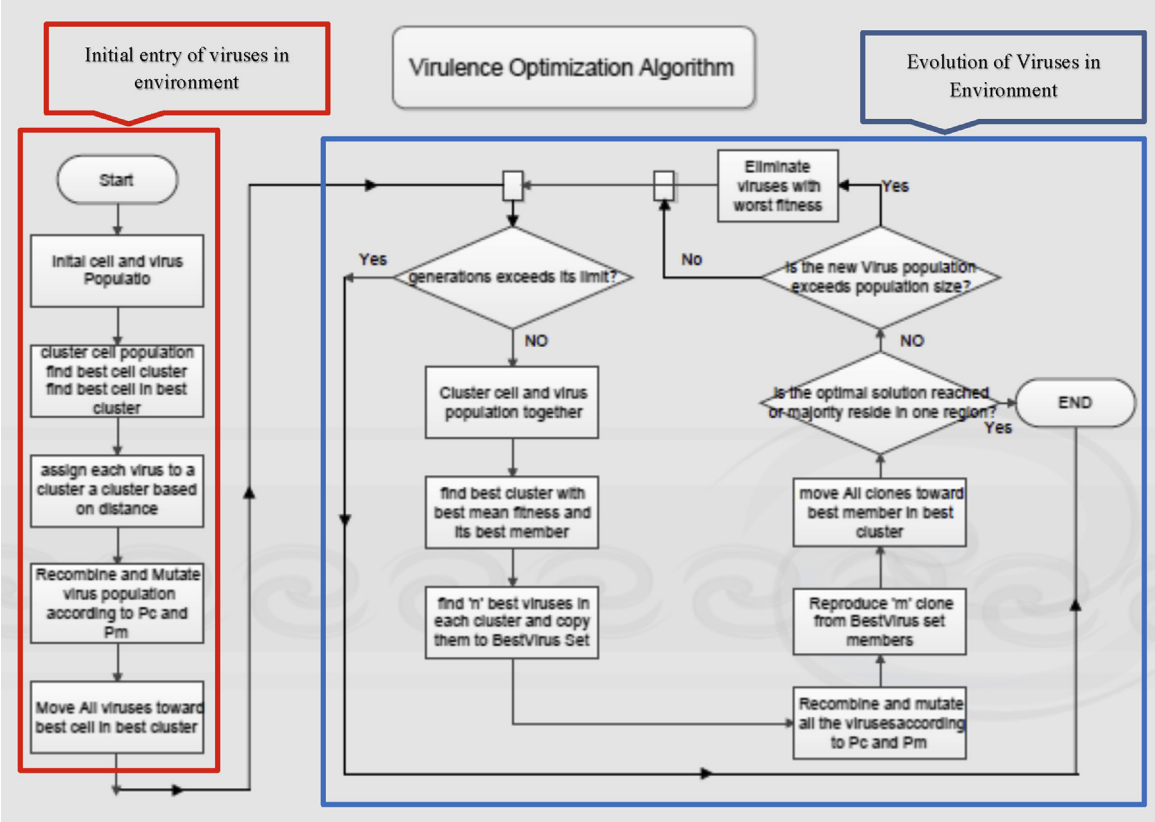

Fig. 5. Flow diagram of the proposed Virulence Optimization Algorithm.

be either in string format, integer number or real-valued number. But for now, we assume that representation of variables is in real-valued numbers.

To measure the amount of profit of environment which viruses reside in or in other words, to measure how much resources vital for survival is in the surrounding region of viruses, we employ a profit function *gene* which its input is the variation of a virus or host cell.

$$f_{gene}(gene\ \ String) = f_{gene}(g_1, \ldots, g_{N_{var}}) \tag{2}$$

The function f_{gene} should be maximized in order to find the global optimum solution. However, since in optimization problem we usually try to minimize the cost, we can modify the profit function as follows to obtain cost function which should be minimized:

$$cost(gene\ \ string) = -f_{gene}(gene\ \ String) = -f_{gene}(g_1, \ldots, g_{N_{var}}) \tag{3}$$

At the initial stage of the virulence optimization algorithm an $N_{var}x\ N_{pop}$-dimensional matrix of gene strands will be generated. This matrix contains the candidate solutions of optimization problem. To generate this matrix, we simply initialize the variable values of each gene strand with a randomly generated number. As mentioned earlier, the search space of virulence algorithm consists of two different populations:

(1) Population of viruses which infiltrate the host environment and try to infect host cells.
(2) Population of host cells, which is affected by virus population.

For each one of these populations, an $N_{var}x\ N_{pop}$-dimensional matrix containing gene strands is generated. It should be noted that, in the first entry of viruses in host environment in order not to raise any suspicion or trigger the immune response, only a small number of viruses enter the host environment. So, in order to emulate this feature of virus varieties, at the initial state of the system the host cell population will be much larger than virus population. During the optimization process, viruses rapidly scatter in host environment and spread the virulence. So, according to the aforementioned fact, we choose the initial host cell population size to be a value from the interval [50,150] and consequently we choose the initial virus population size to be a value from the interval [5,15] since it should a much smaller number than host cell population size. These values operate as the lower and upper bound for virus and cell populations.

To initialize the variables of gene strands we can use "Bone Marrow" algorithm which functions similar to the one in the artificial immune system algorithm. In "Bone Marrow" algorithm for every variable in problem's search space a specific gene library is assigned. These gene libraries are initialized with permissible values of their respective variables (compliance with restrictions of each variable). Finally, during the initialization of each variable, a value is randomly selected from its respective gene library and the variable is initialized with the selected value. Such a structure for initializing variables allows us to assign different properties to each host and virus cell.

3.2. Mutation (Drifting) and recombination (Shifting) as the main driving force behind virulence

As mentioned earlier, the driving force behind the variations in proposed algorithm is the mutation and recombination of virus varieties. Such a property in the virus life allows them to quickly

adapt themselves to the changing circumstances of the host environment and produce different types of virus varieties. In the assortment of segments, viruses can exchange their genetic material to produce new virus variety. The genetic changes that occur in new generated viruses works as a coping mechanism against environmental changes. Proof of existence for such a reproduction in virus life is the observed similarities between different virus varieties in different periods of their genesis cycle. To recombine gene strands of viruses, we can use standard recombination operators such as 1-point Recombination, n-point Recombination or Uniform Recombination. The best performance of virulence algorithm is achieved using Uniform Recombination.

The recombination (Shifting) operation of the virus population allows them, regardless of which region of search space they are in, to search the host environment for the necessary resources necessary for survival and reproduction in the best possible manner (searching for global optimum solution). The recombination of viruses in host environment is controlled "probability of recombination" parameter or P_{rec}. The best observed value for P_{rec} is "0.75". In other words, 75 percent of viruses is recombined in each cycle to produce a new virus variety and efficiently look for optimum solutions.

Virus mutation occurs through random and severe changes in genetic strands. Mutation allows viruses to quickly adapt themselves to changes in the environment and avoid the immune response. In other words, the most important driving force behind the variation in viruses is mutation. Proof of this claim is the resistance of viruses like influenza and its different varieties against antiviral drugs designed to eliminate them.

Spontaneous virus mutation due to rapid growth and reproduction is a key factor in their resistance against an immune response. To mutate viruses, standard mutation (Drifting) operators such as uniform mutation or non-uniform mutation can be employed. It should be noted that the mutation operator should have a varying mutation step size like:

$$\sigma(t)=\begin{cases}\sigma(t-n)*\exp(gen\_bou)/\log(gen\_suc) & \text{if consecutive improvement occurs}\\ \sigma(t-n) & \text{otherwise}\end{cases} \tag{4}$$

Changes in σ are based on feedback from the search operation progress in the search space. The parameter *gen_suc* is the number of generations that improvement happens and its value changes during search progress and *gen_bou* which is generation boundary for evaluating the success of algorithm for a specified problem is set to 10. It should be noted that the varying mutation step size considered for the mutation (Drifting) operators plays an important role in producing new and fitter virus varieties. As the result, this feature is considered to be one of the most vital components of the proposed algorithm. The best performance of virulence algorithm is achieved using Uniform mutation. The main feature of mutation in virus varieties is that allows them to randomly search different regions of the solution space for the region with maximum resources. On the other hand, the mutation (Drifting) operator increases the diversity of candidate solutions in problem's search space. Mutation of viruses in the environment is controlled by "probability of mutation" parameter or P_{mut}. Since viruses have a high mutation rate, unlike other evolutionary algorithms we set P_{mut} to be a large value between "0.1" and "0.3". In other words, 10–30 percent of viruses in the environment will genetically mutate.

3.3. Virus reproduction (cloning) towards virulence

One of the main features of virus life is their excessive and rapid reproduction in the search space. This kind of functionality is achieved through "cloning" operation of viruses. Therefore, one of the major features of the proposed virulence algorithm is to simulate the cloning process. Moreover, the cloning process facilitates the escape mechanism. Cloning serves two major purposes in virus life:

(1) The task of scattering viruses in host environment is the responsibility of cloning. Generally, generating virus clones allows them to rapidly spread in host environment and expand the virulence.
(2) This mechanism allows mature viruses to rapidly clone themselves in the environment and quickly escape to other regions before they get detected and become the subject of immune response.

The details surrounding the "escape" operation will be discussed shortly. It is important to mention that the viruses with higher fitness have a greater chance of cloning themselves and successfully escaping from infected area. Viruses with lower fitness will be detected quickly by the immune system and will be eliminated.

Therefore, we can state that the majority of virus population, which participate in cloning operation are viruses with higher fitness. "CloneNum" parameter controls the frequency of virus cloning. The best observed value for CloneNum parameter in conducted experiments is 10. It means that in each cycle 10 copies of each qualified virus is made and this newly generated viruses will escape to regions with better conditions for growth and reproductions so they do not become the subject of immune response. The higher the fitness of a virus is, the more chance it has to participate in cloning operation and the more the viruses are cloned and reproduced the more infection they can cause in the host environment. During the course of evolution, the cloned viruses are usually scattered around their parent's region in the host environment (corresponding region of each virus is recognized through *K-means* clustering which is described in Section 3.4). If we consider the lower and upper boundary of optimization problem as Low_{Var} and $High_{var}$, each cloned virus will reside in *the Vicinity Region* of their respected parent's region which has a *Vicinity Region Radius*.

$$\begin{aligned}&Vicinity\ \ Region\ \ Radius=(Highvar-LowVar)*\left(\frac{number\ \ of\ \ clones\ \ (CloneNum)}{Total\ \ number\ \ of\ \ viruses\ \ in\ \ the\ \ region+1}\right)^{\beta}\\ &if\ \ 0<\left(\frac{number\ \ of\ \ clones\ \ (CloneNum)}{Total\ \ number\ \ of\ \ viruses\ \ in\ \ the\ \ region}\right)\le 1\ \ then\ \ \beta\ \ is\ \ a\ \ random\ \ number\ \ between\ \ 1\ \ and\ \ 100\\ &otherwise\ \ \beta\ \ is\ \ a\ \ random\ \ number\ \ between\ \ 0\ \ and\ \ 1\end{aligned} \tag{5}$$

3.4. Escape towards best living area (maximum resource)

The most vital component of the proposed algorithm is the escape mechanism which contributes enormously to the novelty of this approach. As mentioned earlier, during the evolutionary process, there will be a selection in favour of those varieties of viruses which vegetate in host environment whence they can escape, so they do not become subject of immune response. The escape process brings two important features in the life cycle of viruses:

(1) Allows the mature viruses which have higher fitness compared to others to efficiently reproduce and infect the host environment. In other words, allows the viruses to search better for regions with maximum resource in candidate solution space. As the result of this process, the virulence of viruses in environment highly increases.
(2) It does not let the mature viruses which are cloning and infecting the host environment to be detected by the immune system. Because as soon as a mature virus grew, reproduced and mutated the appropriate amount, it escapes the infected region to a region of environment with higher chance of survival and more resources. From another point of view, it could be said that when viruses reside in a specific region and infect it, they use up all the resources in that region. When the resources are on the verge of ending, the immune system becomes aware of foreign invader's existence and readies itself to retaliate. Viruses with higher fitness compared to other viruses, rapidly reproduce themselves and before they become the subject of immune response, they escape the infected region to a region of environment with higher chance of survival and more resources.

It should be noted that when viruses reside in any region of environment they tend to form virus groups or societies. When it is the time to escape towards better regions, the group (society) of viruses which have the highest mean fitness compared to other groups will be selected as the escape destination. But since the viruses are scattered all over the host environment, it is difficult to recognize which virus belongs to which group. To solve this problem, the grouping of viruses is done using K-means clustering method (the value of 3 for **k** parameter seems to be sufficient). Now that the virus groups are constituted the mean fitness value of each group is calculated. Then the group with highest mean fitness is chosen as escape destination and the virus with the highest fitness in this group is chosen as the escape point which all the mature viruses escape to. The region surrounding this group (group with highest mean fitness) can be considered as the region which is most probable to contain the global optimum solution or maximum resources.

When escaping towards the destination point, the virus does not reside completely and directly near the destination group. They only traverse a part of the way to the destination group and also have a deviation in their path. The mechanism of virus escape towards the selected group is illustrated in Fig. 6.

As it is illustrated in the figure, each mature virus which has high fitness only traverses α % of distance to the destination point and also has a deviation of β radians. These two parameters α and β help the viruses to search more positions around the destination point to find the best region with maximum resources. The parameters α and β are defined as follows:

$$\alpha \sim U(0, 1)$$
$$\beta \sim U(-\varphi, \varphi) \quad (6)$$

The value of parameter α is a random number between 0 and 1 generated by a uniform distribution and β is a parameter which controls the deviation from the optimal point (destination point) which is a random number between $-\varphi$ and φ generated by a uniform distribution. The value of $\pi/2$ for parameter φ seems to be sufficient for good convergence of the virulence algorithm to global optimum solution.

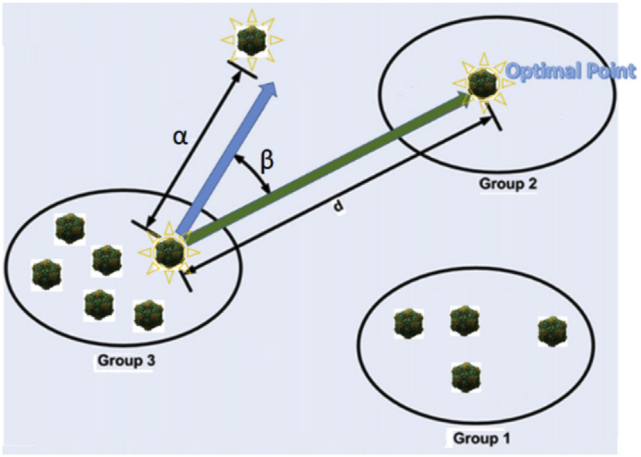

Fig. 6. Virus escape mechanism.

3.5. Eliminating virus with worst fitness

As stated in short-sighted evolution theory (which describes the virus evolution), in a virus population the varieties which have a higher transmission rate and hence higher virulence and fitness, have a higher chance for survival, growth and reproduction. In other words, as the result of the fierce competition between viruses to obtain vital resources and their rapid and excessive reproduction in host environment the body's immune system becomes aware of foreign invader's existence. Through the escape mechanism, the viruses with higher transmission rate, higher virulence level and higher fitness escape to best region containing maximum resources vital for survival. In the meantime, the viruses with lower fitness do not get selected to escape and remain in their respective region. If the fitness of these viruses is very low, they will be detected quickly by the immune system and become subject to immune response and consequently get eliminated.

The elimination of the worst viruses maintains the equilibrium of the host environment. On the other hand, the viruses with worst fitness represent the regions of candidate solutions space which are less likely to contain global optimum solutions.

From another point of view, it could be said that in each iteration of the virulence optimization algorithm the number of viruses in the environment increases quickly. After the escape process, to restore the balance in host environment and to reduce the population to its normal size, a number of viruses which have the worst fitness will be eliminated so the equilibrium is maintained.

3.6. Convergence

Like algorithms such as cuckoo optimization algorithm and ICA, after a certain number of iterations all the viruses in environment will converge to one of the specified virus groups (societies) in host environments. The region surrounding this group (society) has the maximum resources vital for survival, growth and reproduction. Also the viruses in this region have the minimum chance of detection by the body's immune system. In general, one of these termination conditions will terminate the optimization process of virulence algorithm:

(1) The permissible number of iterations for virulence algorithm is reached.
(2) The optimal value of the objective or profit function is reached.
(3) The majority of viruses in host environment are converged to a specific point in candidate solutions space.

The pseudo code for the virulence optimization algorithm is illustrated in Fig. 7.

1. *Initial virus and cell population using bone marrow algorithm and evaluate them.*

A. ***Initial Entrance of viruses in cells' environment***:

2. *Cluster cell population, find best cluster and set best cell in that cluster as escape point.*
3. *Assign each virus to a cluster based on the distance.*
4. *Recombine and mutate virus population according to P_{rec} and $P_{mut.}$*
5. *Move all viruses toward escape point (escape) and evaluate.*

A*. ***End of Initial Entry***.

B. ***Virulence Stage:*** *if generation numbers exceeds stop, O.W. continue:*

6. *Cluster Cell and Virus together, find best cluster with best mean fitness and set its best member as escape point.*
7. *Find n best virus and n' random virus in each cell copy them to BestVirus Set.*
8. *Recombine and mutate all viruses according to P_{rec} and $P_{mut.}$*
9. *Produce m clone from BestVirus Set.*
10. *Move all clones toward escape point (escape) and Evaluate.*
11. *If Optimal Solution is reached or vast majority of viruses reside in one region, Stop, O.W. Continue.*
12. *If the Virus Population Exceeds allowed number, Eliminate Viruses with Worst Fitness, else go to B.*

Fig. 7. The pseudo code for virulence optimization algorithm.

The strength of evolutionary algorithms with stochastic behaviour such as this proposed algorithm stems from the fact that their probabilistic and somewhat random structure ensures that this algorithm will not necessarily gets stuck at local optima, and if it got stuck in local optima, the embedded variation mechanisms such as mutation and escape allows the algorithm to escape from local optima.

4. Evaluation and benchmarking on Virulence Optimization Algorithm

In this section, the proposed virulence optimization algorithm is evaluated using 11 test function. These functions are designed in such a way so they can evaluate the capability of optimization algorithms in tackling obstacles such as discontinuities and multi-modality in reaching optimal solution. Five of these test functions are De Jong function [39–41], five of them are from Ref. [3] and a 10-dimensional Rastrigin function [42,43].

4.1. Test objective functions (cost function) for evaluation

All the functions used for the evaluation of optimization algorithm in this section are cost functions which transform the optimization problem into a minimization one:

Function 1. First De Jong function is a continuous, convex and Uni-modal function. The function is usually evaluated on the hyper-cube $x_i \in [-5.12, 5.12]$, for all $i = 1, 2, \ldots, n$. This function is also called sphere function.

$$f(x) = \sum_{i=1}^{n} x_i^2 \quad minimum : f(0, \ldots, 0) = 0 \tag{7}$$

Function 2. Rosenbrock valley is one of the classic optimization problems which is also called the Banana function or second De Jong function. Finding the valley is easy, however, convergence to the global optimum is difficult and hence this problem has been repeatedly used in assessing the performance of optimization algorithms. The function is usually evaluated on the hyper-cube $x_i \in [-2.048, 2.048]$, for all $i = 1, 2, \ldots, n$.

$$f(x) = \sum_{i-1}^{n-1} \left[100(x_{i+1} + x_i^2)^2 + (1 - x_i)^2 \right] \quad minimum : f(1, \ldots, 1) = 0 \tag{8}$$

Function 3. The third De Jong function is a non-convex, Uni-Modal and n-dimensional function. This function is used to evaluate the performance of an algorithm in discontinuous problem. The function is usually evaluated on the hyper-cube $x_i \in [-5.12, 5.12]$, for all $i = 1, 2, \ldots, n$.

$$f(x) = \sum_{i=1}^{n} |x_i| \quad \min imum : f(0, \ldots, 0) = 0 \tag{9}$$

Function 4. The fourth De Jong function is a continuous, convex, Uni-Modal and n-dimensional function and usually is defined along with a Gaussian noise to evaluate the capability of the algorithm in the face of noisy feature space. The function is usually evaluated on the hyper-cube $x_i \in [-1.28, 1.28]$, for all $i = 1, 2, \ldots, n$.

$$f(x) = \sum_{i=1}^{n} i x_i^4 + Gaussian \quad Noise \quad \min imum : f(\sim) = \sim 0 \tag{10}$$

Function 5. The fifth De Jong function is an interesting multi-modal function which is designed by shekel in 1971. This function is a continuous, non-convex and bi-dimensional one that contains

25 local optimum in points $\{(a_{1j}, a_{2j})\}_{j=1}^{25}$. The function is usually evaluated on the square $x_i \in [-65.536, 65.536]$, for $i = 1, 2$.

$$(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$$
$$f(x_1, x_2) = \left\{ 0.002 + \sum_{j=1}^{25} [j + (x_1 - a_{1j})^6 + (x_2 - a_{2j})^6]^{-1} \right\}^{-1} \quad (11)$$

This function is also called the shekel of FoxHole function. This function contains a lot of local optimums and many standard optimization algorithms get stuck in the first found peak.

Function 6. In mathematical optimization, the Rastrigin function is a non-convex function used as a performance test problem in optimization algorithms. It is a typical example of non-linear multimodal function. It was first proposed by Rastrigin [42] as a 2-dimensional function and has been generalized by Mühlenbein et al. [43]. Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima [42]. The function is usually evaluated on the hyper-cube $x_i \in [-5.12, 5.12]$, for all $i = 1, 2, \dots, n$.

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)), \quad n = 10 \quad \min imum : f(\sim) = 0 \quad (12)$$

Function 7. The Griewank function is very similar to Rastrigin function. The difference between this function and Rastrigin is that the local optimums of this function are widely scattered in problem's search space. The function is usually evaluated on the hyper-cube $x_i \in [-600, 600]$, for all $i = 1, 2, \dots, n$.

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad \min imum : f(\sim) = 0 \quad (13)$$

Function 8. The Ackley's path function is one of the popular test functions in evaluating optimization algorithms. The function is usually evaluated on the hyper-cube $x_i \in [-32.768, 32.768]$, for all $i = 1, 2, \dots, n$. This function is defined as follows.

$$f(x) = -a \cdot e^{-b \cdot \sqrt{\sum_{i=1}^{n} x_i^2 / n}} - e^{\sum_{i=1}^{n} \cos(c \cdot x_i)/n} + a + e^1 \quad a = 20, \quad b = 0.2, \quad c = 2 \cdot pi \quad (14)$$
$$\min imum : f(\sim) = 0$$

Function 9. This function is a bi-dimensional function of the following form. The function is usually evaluated on the square $x_i \in [-\infty, \infty]$, for all $i = 1, 2$.

$$f(x, y) = (x^2 + y^2)^{0.25} * \sin\{30[(x + 0.5)^2 + y^2]^{0.1}\} + |x| + |y|,$$
$$\min imum : f(-0.2, 0) = -0.247 \quad (15)$$

Function 10. This function is a bi-dimensional function of the following form. The function is usually evaluated on the square $x_i \in [-100, 100]$, for all $i = 1, 2$.

$$f(x, y) = x * \sin(4x) + 1.1y * \sin(2y),$$
$$\min imum : f(9.039, 8.668) = -18.5547 \quad (16)$$

Function 11. The Schaffer function is a bi-dimensional function of the following form. The function is usually evaluated on the square $x_i \in [-10, 10]$, for all $i = 1, 2$.

Table 3
The parameter settings of the designed tests.

Parameters	Setting #1	Setting #2
Cell population	100	1000
Virus population	10	100
Mutation (Drift) prob.	0.1	0.1
Reproduction (Shifting) prob.	0.75	0.75
Clusters no.	3	5
Clone no.	10	10
No. of variables	10	100
Lower bound.	−10	−25
Upper bound.	10	25

$$f(x) = 0.5 + \frac{\sin^2(x_1^2 + x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2},$$
$$\min imum : f(0, 1.25313) = 0.0015685 \quad (17)$$

4.2. Evaluation results

To evaluate the proposed algorithm, two different sets of tests with different parameters are considered. The tests are designed to test the various aspects, the convergence speed, and the required operational time of the proposed evolutionary algorithm. The first set evaluates the ability of the proposed algorithm to solve standard optimization problems and generate an optimal solution in terms of the convergence speed and the required operational time. The second set evaluates the ability of the proposed algorithm to tackle high-dimensional optimization problems. Table 3 illustrates the parameter settings of each test.

4.2.1. Evaluation results of the first test

To compare the performance of the proposed algorithm with other well-known optimization algorithms, we have evaluated the test object functions using PSO algorithm, cultural algorithm, cuckoo optimization algorithm (COA) and genetic algorithm (GA). Table 4 illustrates the Parameter settings of the well-known optimization algorithms used for comparison in the first test.

At first, we evaluate the proposed virulence optimization algorithm with 6th, 9th and 11th function. We start with Rastrigin function which has a global optimum solution at the point $(x_1, x_2, \dots, x_n) = (0, 0, \dots, 0)$ with the value 0. The 3-dimensional plot of this function is illustrated in Fig. 8a.

To find the global optimum solution of the Rastrigin function via virulence optimization algorithm, the following values for algorithm parameters are considered:

The initial size of the body cell population is set to 100 and the initial size of virus population is set to 10. The number of cloning performed on each qualified virus in each cycle is set to 10. In each iteration, the virus population is clustered into 3 groups using the K-means algorithm. The length of each gene strand is set to 10 and the upper-bound and lower-bound values of each virus and cell is set to (−10) and (+10) respectively. The probability of recombination and mutation is set (0.75) and (0.1) respectively.

As illustrated in Fig. 8b, the virulence algorithm is able to find the global minimum of Rastrigin function in 29 iterations. As it is visible, on 10th iteration of the algorithm the majority of the virus

Table 4
the Parameter settings of the well-known optimization algorithms in the first test.

COA						
Parameter	Population	Max. eggs	Min. eggs	Clusters	No. of variables	Boundary
Value	100	10	2	5	10	[−10,10]

PSO					
Parameter	Population	Social constant	Cognitive constant	No. of variables	Boundary
Value	100	2	2	10	[−10,10]

Cultural						
Parameter	Population	Alpha	Beta	Acceptance rate	No. of variables	Boundary
Value	100	0.25	0.5	0.35	10	[−10,10]

Genetic algorithm					
Parameter	Population	Crossover prob.	Mutation prob.	No. of variables	Boundary
Value	100	0.75	0.1	10	[−10,10]

Virulence Optimization Algorithm						
Parameter	Cell and virus population	Mutation (Drift) prob.	Reproduction (Shifting) prob.	Clusters and clone no.	No. of variables	Boundary
Value	100 and 10	0.1	0.75	3 and 10	10	[−10,10]

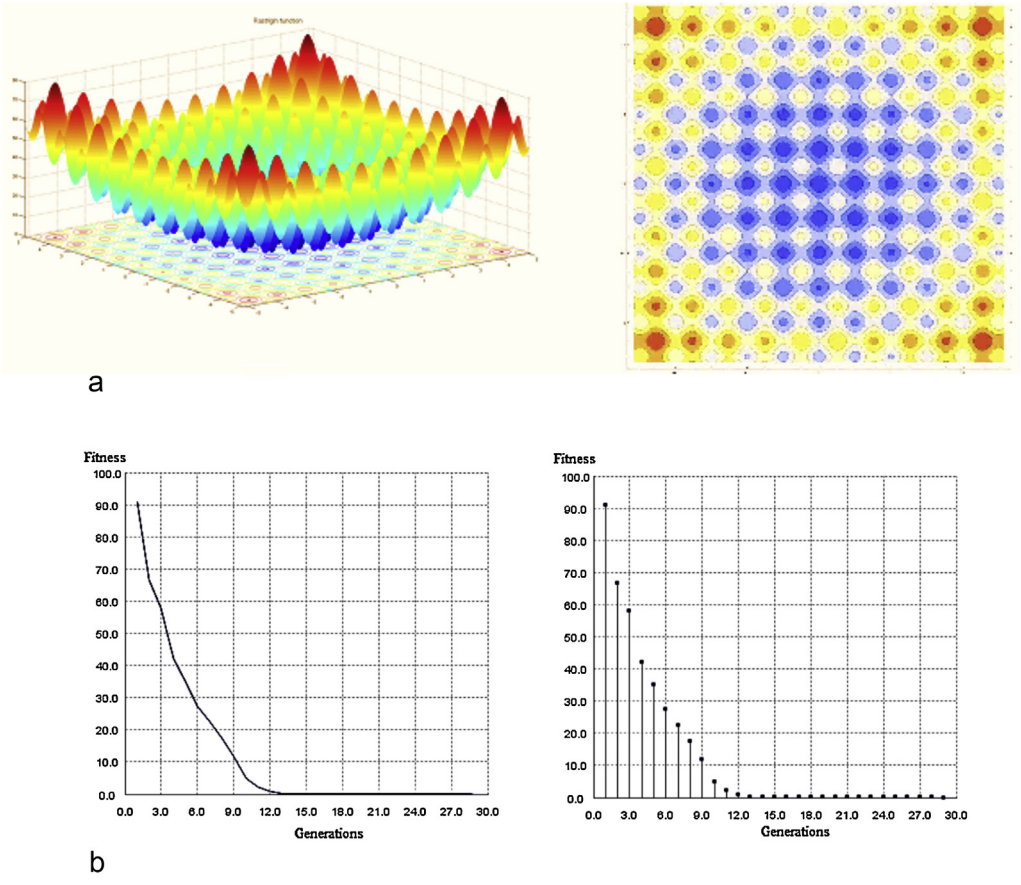

Fig. 8. (a) (3-Dimensional) plot of Rastrigin function. (b) Convergence of virulence algorithm on the Rastrigin function.

population in host environment reaches the region which contains the global minimum solution. During the subsequent iterations, viruses are searching more positions near the global minimum region until in 29th iteration the virulence algorithm converges to the global minimum point. The coordinates of global minimum is equivalent to $(x_1, x_2, \ldots, x_n) = (0,0,\ldots,0)$ and its value is (0).

The 9th function is a bi-dimensional one which has the global minimum of $(x, y) = (-0.2, 0)$ with value of (-0.247). The 3-dimensional plot of this function is illustrated in Fig. 9 [3]. To find the global optimum solution of this function via virulence optimization algorithm we used the same configuration as the previous experiment.

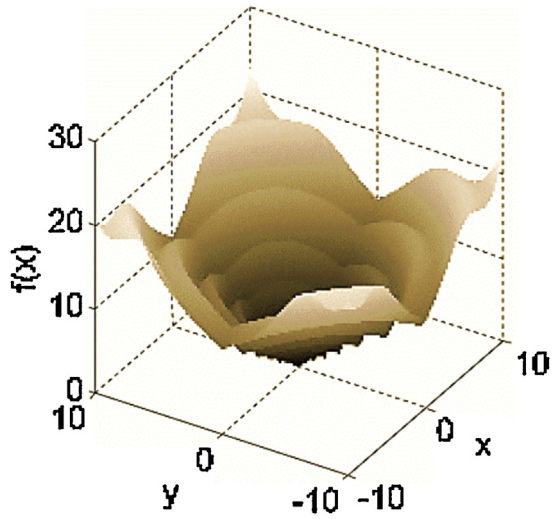

Fig. 9. (3-Dimensional) plot of the 9th function.

As illustrated in Fig. 10, the virulence algorithm is able to find the global minimum of this function in 13 iterations. As it is visible, on 10th iteration of the algorithm the majority of the virus population in host environment reaches the region which contains the maximum resources or a global minimum solution or minimum cost. During the subsequent iterations and in 29th iteration the virulence algorithm converges to the global minimum point with the cost value of (−0.2471802).

The 11th function or the Schaffer function is a bi-dimensional one which has the global minimum of $(x, y) = (0, 1.25313)$ with value of (0.0015685). The 3-dimensional plot of this function is illustrated in Fig. 11 [3]. To find the global optimum solution of this function via virulence optimization algorithm we used the same configuration as the previous experiments.

As illustrated in Fig. 12, the virulence algorithm has been able to find the global minimum of this function in 9 iterations. As it is visible, on 9th iteration of the algorithm the majority of the virus population in host environment reaches the region which contains the global minimum solution or minimum cost and the virulence algorithm converges to the global minimum point with the cost value of (0.001568558).

To compare the performance of the proposed algorithm with other well-known optimization algorithms, we evaluated the 6th, 9th and 11th function using PSO algorithm, cultural algorithm, cuckoo optimization algorithm (COA) and genetic algorithm (GA) with roulette wheel selection and Uniform Crossover.

Due to the fact that different initial population size for each method affects directly the result and the speed and performance of algorithms, a series of 10 test runs per each algorithm is done and then the mean expectance of performance for each method is calculated and considered as the comparison criteria.

At first, we apply these four algorithms on the Rastrigin function (6th function). We set the permissible number of iterations to 1000 because converging to the global minimum of this function

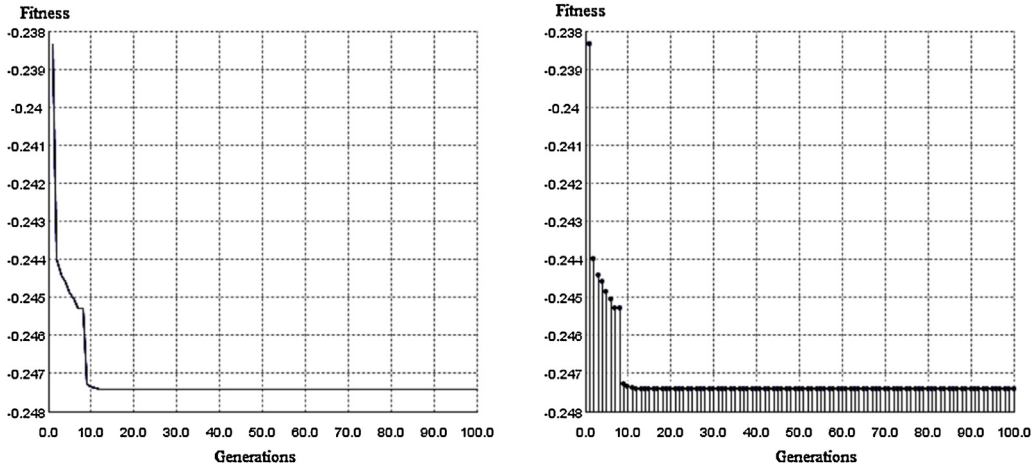

Fig. 10. Convergence of virulence algorithm on the 9th function.

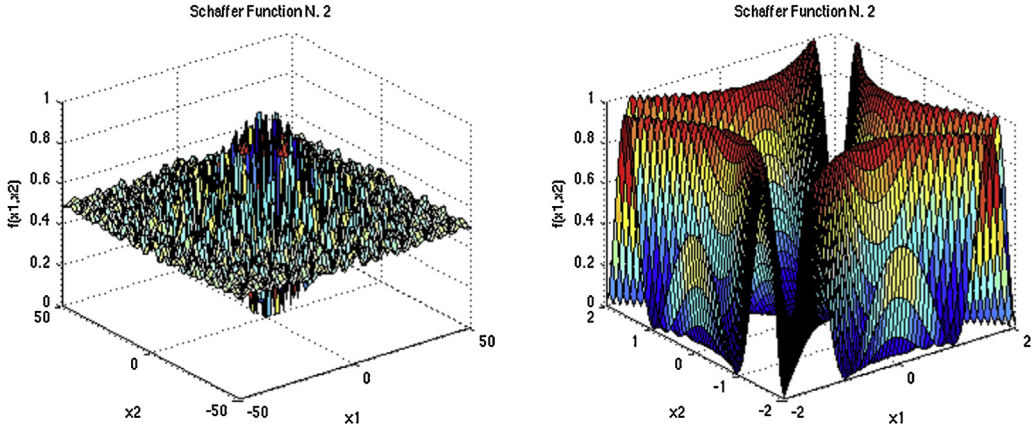

Fig. 11. (3-Dimensional) plot of the Schaffer function.

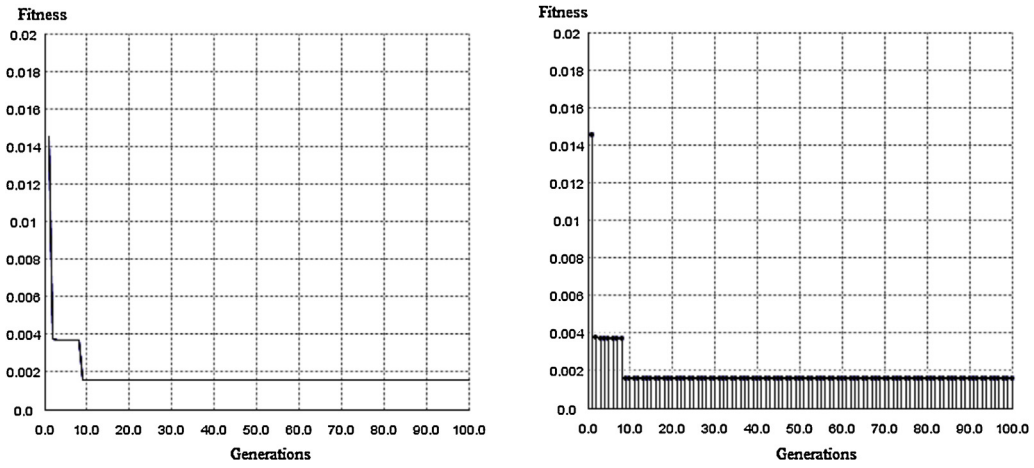

Fig. 12. Convergence of virulence algorithm on the Schaffer function.

is fairly difficult for optimization algorithms. Running the simulations on this function produces a mean of 28.6, 178.9, 263.8, 54.6 and 1000 as the stopping iterations to reach convergence for virulence, GA, Cuckoo, PSO and cultural algorithms. It should be noted that from the 10 test runs performed using each algorithm, the virulence algorithm reached the global optimum 10 times, GA reached the global optimum 0 (zero) time, the Cuckoo reached the global optimum 9 times, the PSO reached the global optimum 10 times and the cultural reached the global optimum 0 (zero) time. As it is evident from these results the virulence algorithm not only reached the global minimum in all the test runs, but also did it in fewer iterations compared to other algorithms. The PSO algorithm achieved the second best results on the Rastrigin function and the Cuckoo algorithm is the third.

Figs. 13–16 depict a single sample of experiments done on GA, Cuckoo, PSO and cultural using the Rastrigin function (6th function) respectively. As it is seen from the figures, the GA reached a non-optimum solution after 257 iterations, the Cuckoo reached the optimum solution after 171 iterations, the PSO reached the optimum solution after 61 iterations and the cultural algorithm reached a non-optimum solution after 79 iterations.

So far, the virulence optimization algorithm has been able to outperform the other well-known algorithms in the optimization of Rastrigin function. It should be noted that the Rastrigin function is a complex one and reaching its global optimum point is fairly difficult. However, the proposed algorithm has converged to its global optimum point in fewer iterations compared to other algorithms. It can be concluded that the proposed algorithm achieves faster convergence. It should also be noted that a number of algorithm used in this experiment are not even able to converge to global optima or they converge in more iterations than the proposed algorithm.

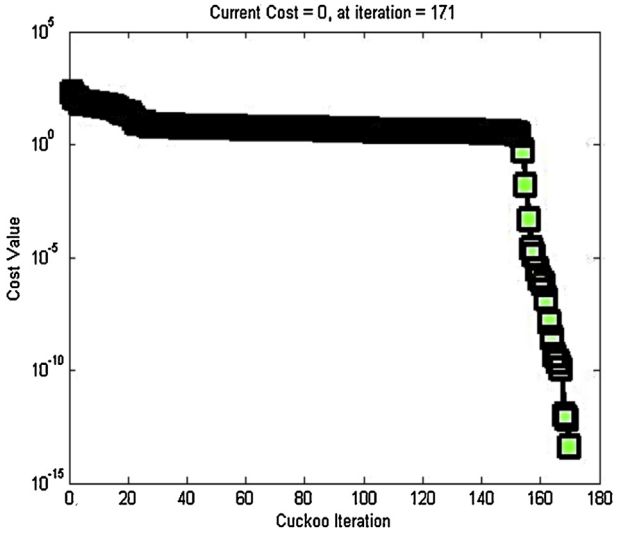

Fig. 14. Cost minimization using COA.

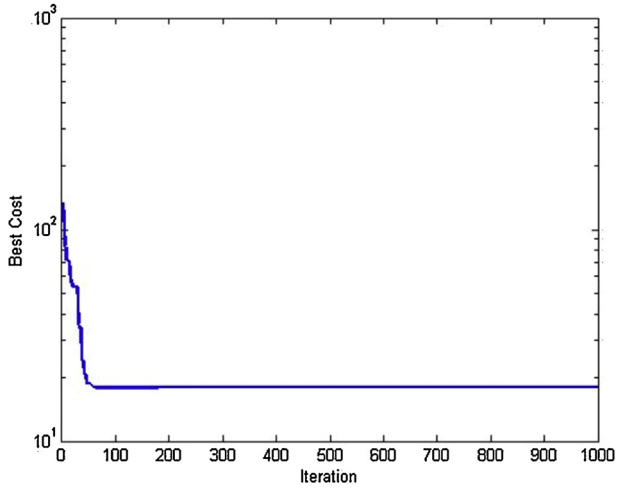

Fig. 13. Cost minimization using cultural.

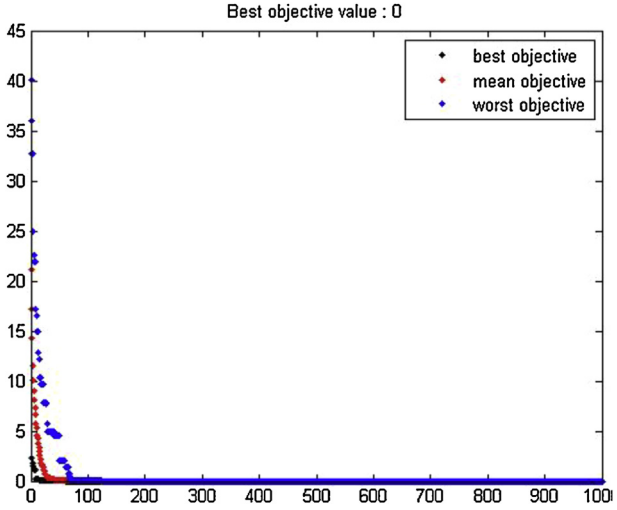

Fig. 15. Cost minimization using PSO.

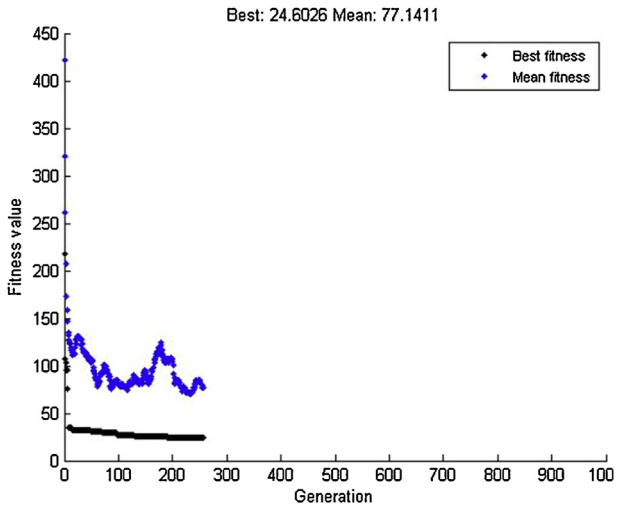

Fig. 16. Cost minimization using GA.

Now we apply these four algorithms on the 9th function. We set the permissible number of iterations to 1000. However, during the optimization process, we realized that 100 iterations are sufficient to reach convergence using the 9th function. Running the simulations on this function produces a mean of 12.4, 128.3, 23.7, 39.5 and 36.7 as the stopping iterations to reach convergence for virulence, GA, Cuckoo, PSO and cultural algorithms. It should be noted that from the 10 test runs performed using each algorithm, the virulence algorithm reached the global optimum 10 times, GA reached the global optimum 2 times, the Cuckoo reached the global optimum 7 times, the PSO reached the global optimum 2 times and the cultural reached the global optimum 5 times. As it is evident from these results the virulence algorithm not only reached the global minimum in all the test runs, but also did it in fewer iterations compared to other algorithms. The Cuckoo algorithm achieved the second best results on the 9th function and the cultural algorithm is the third.

Figs. 17–20 depict a single sample of experiments done on GA, Cuckoo, PSO and Cultural using the 9th function respectively. As it is seen from the figures, the GA reached a non-optimum solution after 155 iterations, the Cuckoo reached the optimum solution after 27 iterations, the PSO reached the optimum solution after 41 iterations and the Cultural algorithm reached a non-optimum solution after 79 iterations.

In this experiment the virulence optimization algorithm has been able to outperform the other well-known algorithms in the optimization of 9th function. Moreover, the virulence algorithm is the only algorithm that has been able to converge to the optimum solution in all the 10 test runs and has done it in the least iterations possible. In other words, it achieves faster convergence than the other algorithms.

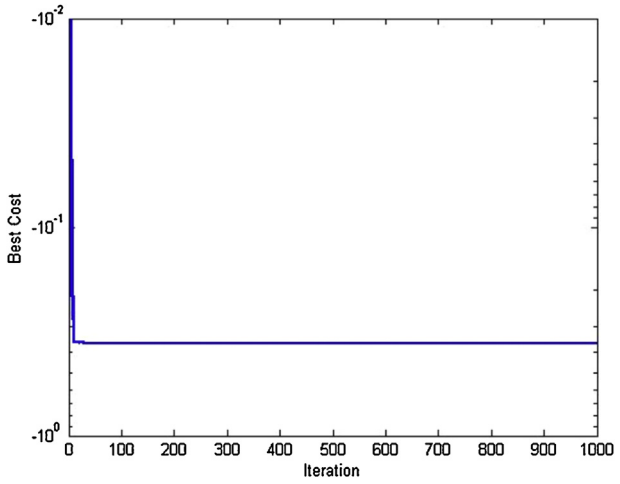

Fig. 17. Cost minimization using cultural.

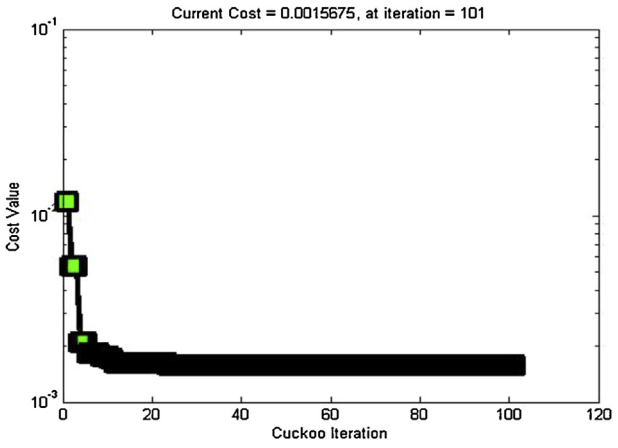

Fig. 18. Cost minimization using COA.

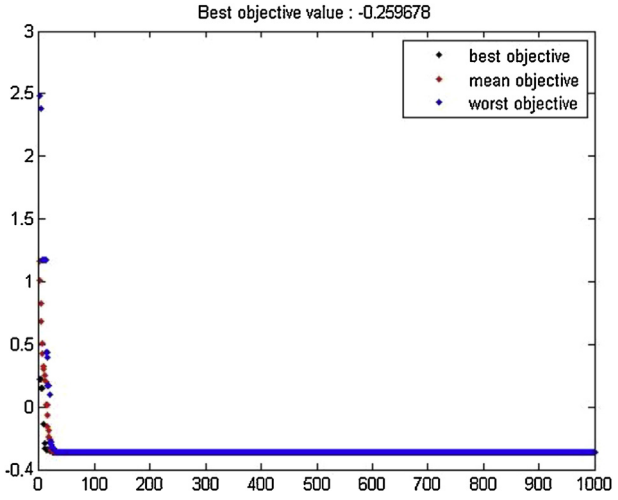

Fig. 19. Cost minimization using PSO.

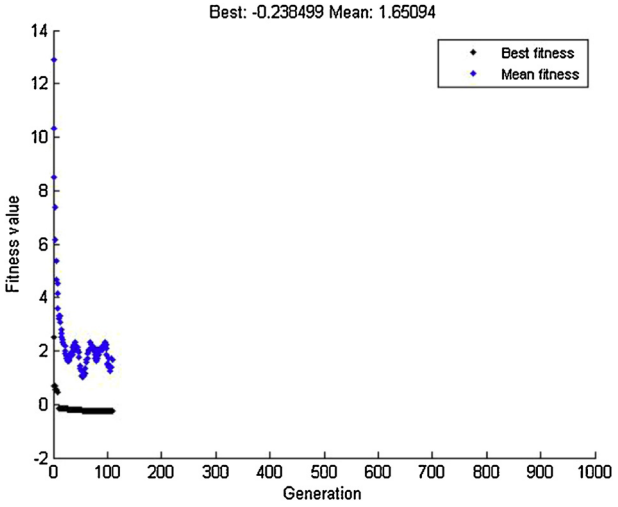

Fig. 20. Cost minimization using GA.

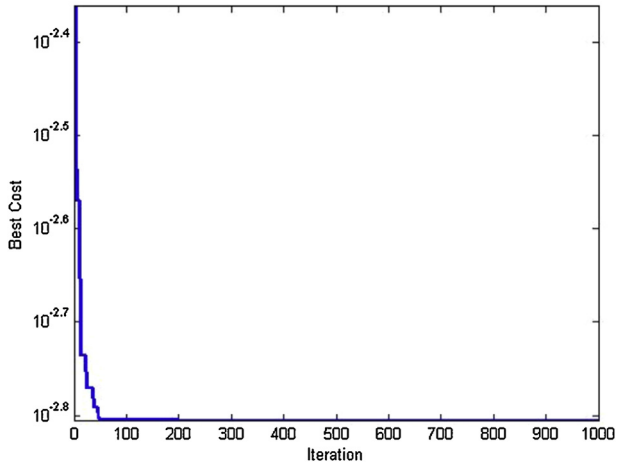

Fig. 21. Cost minimization using cultural.

Next we apply these four algorithms on the Schaffer function (11th function). We set the permissible number of iterations to 100 because in the majority of studied articles, the optimization algorithms have converged to the global optimum solution of this function in 100 iterations. Running the simulations on this function produces a mean of 8.75, 51, 30.2, 18.2 and 47.6 as the stopping iterations to reach convergence for virulence, GA, Cuckoo, PSO and cultural algorithms. It should be noted that from the 10 test runs performed using each algorithm, the virulence algorithm reached the global optimum 10 times, GA reached the global optimum 6 times, the Cuckoo reached the global optimum 10 times, the PSO reached the global optimum 10 times and the cultural reached the global optimum 10 times. As it is evident from these results all the algorithm except GA converged to the global minimum in all the test runs. Again the virulence algorithm has converged to the global optimum solution in fewer iterations compared to other algorithms. The PSO algorithm achieved the second best results on the Schaffer function and the Cuckoo algorithm is the third. It be noted is that although the PSO algorithm demonstrates similar performance to the virulence algorithm, the virulence algorithm converges to the global optimum solution in a mean of 8.75 (approximately 9) iteration.

Figs. 21–24 depict a single sample of experiments done on GA, Cuckoo, PSO and cultural using the Schaffer function respectively. As it is seen from the figures, the GA reached a non-optimum solution after 51 iterations, the Cuckoo reached the optimum solution

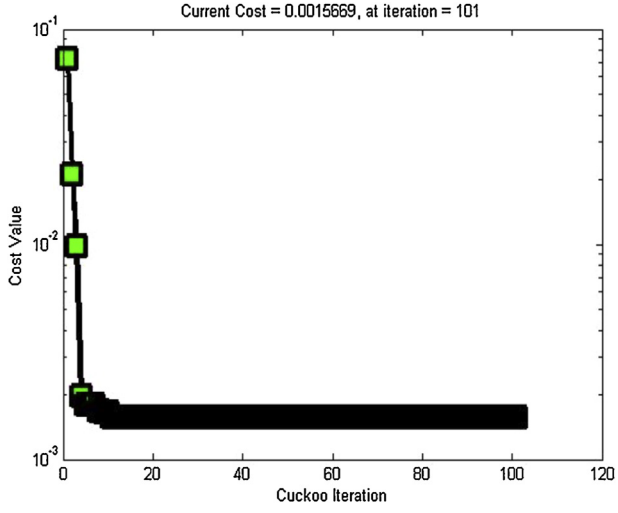

Fig. 22. Cost minimization using COA.

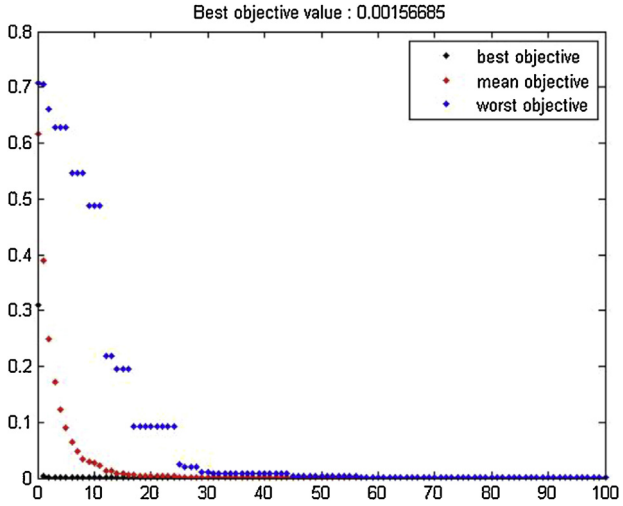

Fig. 23. Cost minimization using PSO.

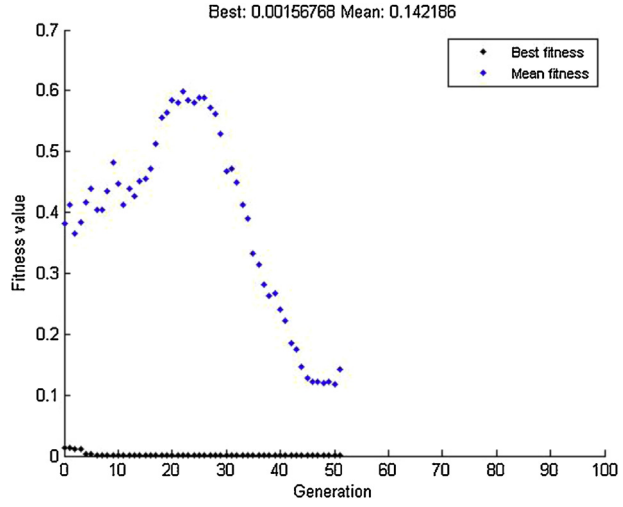

Fig. 24. Cost minimization using GA.

after 39 iterations, the PSO reached the optimum solution after 17 iterations and the cultural algorithm reached the optimum solution after 56 iterations.

So far and according to the results of these experiments, it can be concluded the virulence algorithm outperformed the other optimization algorithms. The most important feature of this algorithm is its fast convergence. The summary of the 3 conducted experiments' results is illustrated in Table 5.

To conduct more experiments, we evaluate the virulence algorithm and the other four algorithms using 1st, 2nd, 3rd, 4th, 5th,

Table 5
the mean stopping iterations required for convergence in 10 test runs and the number of convergence to the global optimum solution occurred in 10 test runs.

	Rastrigin function	9th function	Schaffer function
GA	178.9	128.3	51
No. of convergence	Zero	2	6
PSO	54.6	39.5	18.2
No. of convergence	10	2	10
Cuckoo	263.8	23.7	30.2
No. of convergence	9	7	10
Cultural	1000	36.7	47.6
No. of convergence	Zero	5	10
Virulence	28.6	12.4	8.7
No. of convergence	10	10	10

Table 6
Convergence value and the iterations needed to converge after 1000 iterations (virulence).

Virulence	F1	F2	F3	F4	F5	F7	F8	F10
Convergence value	0.0	0.0064138	0.0	0.0	0.0	0.0	4.4048E−16	1.32E−308
Iterations	382	1000	713	193	82	24	56	720

Table 7
Convergence value and the iterations needed to converge after 1000 iterations (GA).

GA	F1	F2	F3	F4	F5	F7	F8	F10
Convergence value	0.276247	108.65	1.07543	0.204851	12.6705	0.000326	2.63827	−16.776
Iterations	78	51	65	51	51	59	78	63

Table 8
Convergence value and the iterations needed to converge after 1000 iterations (Cuckoo).

Cuckoo	F1	F2	F3	F4	F5	F7	F8	F10
Convergence value	11.9522	0.029774	0.0	0.0	12.6705	0.0	8.8818E-16	−9.3084
Iterations	1000	1000	837	223	1000	24	1000	1

Table 9
Convergence value and the iterations needed to converge after 1000 iterations (PSO).

PSO	F1	F2	F3	F4	F5	F7	F8	F10
Convergence value	4.42E−134	0.0	1.1E−67	3.36E−275	12.6705	0.0	8.8817E−16	−8.34E120
Iterations	1000	208	1000	1000	1000	47	1000	1000

Table 10
Convergence value and the iterations needed to converge after 1000 iterations (cultural).

Cultural	F1	F2	F3	F4	F5	F7	F8	F10
Convergence value	1.59E−102	8.9999	5.007E−51	6.71E−197	12.6705	0.0	4.441E−15	−238.6873
Iterations	1000	1000	1000	1000	1000	73	1000	1000

7th, 8th and 10th test functions. Tables 6–10 depicts the minimization result of virulence, GA, Cuckoo, PSO and cultural algorithm on a random run (the random run for each algorithm is the run that achieves the best result among a set of five random runs). In these runs, the size initial population is set to 100 and the permissible number of iterations is set to 1000. Table 11 illustrates the mean stopping iterations for convergence in 10 test runs for the aforementioned function using all the algorithms.

We now investigate the performance of optimization algorithms on each aforementioned test function.

1st function (F1). As it is seen from above tables, the best result to optimize this function is achieved using the proposed algorithm. The proposed algorithm has been able to converge to the global optimum solution. Although some of the algorithms got really close to the optimal solution, none of them have been able to converge to zero (optimal solution).

2nd function (F2). It is the only function that this proposed algorithm could not perform better than or similar to the others. The best result for optimization of this function is achieved using the PSO algorithm. The proposed algorithm achieved the second best results.

3rd function (F3). Only the proposed algorithm and the Cuckoo algorithm have been able to converge to the global optimum solution, yet the proposed algorithm reached the convergence in fewer iterations than Cuckoo algorithm.

4th function (F4). Again, only the proposed algorithm and the Cuckoo algorithm have been able to converge to the global optimum solution, yet the proposed algorithm reached the convergence in fewer iterations than Cuckoo algorithm.

5th function (F5). The most robust performance of this algorithm is achieved optimizing this function. The proposed algorithm has been able to reach the global optimum solution of zero, whereas the other analogous optimization algorithms merely converged to the value of (12.6705).

7th function (F7). Except the GA, all the other algorithms have been able to reach the global optimum solution of zero. Among them, the Cuckoo and the proposed algorithms have performed similar to each other and have reached the global optimum in equal iterations.

8th function (F8). The results obtained by all the algorithms are fairly similar. However, the proposed algorithm has converged to a better minimum value compared to the other algorithms.

10th function (F10). In this function the proposed algorithm has converged to a better minimum value compared to the other algorithms.

According to these results we can interpret the following items as the strength points and capabilities of virulence algorithm:

1. When most of the times other algorithms have not been able to reach the global optimum, the proposed algorithm has been able to reach it or reach a better minimum value than them. Of course, the F2 or Rosenbrock function is the exception which PSO algorithm converged to the global optimum solution and the proposed algorithm could not reach the region containing global optimal solution.
2. The convergence speed of this algorithm. As it is evident from Tables 2–6, in cases which all or some of the algorithms have converged to similar results, the proposed algorithm is converged in fewer iterations. This is a very important and useful feature for a robust and fast evolutionary algorithm.

In the last stage of evaluation, the time of convergence for each algorithm after 100 iterations is calculated and the results are compared. In other words, all the algorithms are evaluated on all

Table 11
the mean stopping iterations required for convergence in 10 test runs.

	F1	F2	F3	F4	F5	F7	F8	F10
GA	1000	1000	1000	1000	1000	1000	1000	1000
PSO	1000	231.6	1000	1000	1000	78	1000	1000
Cuckoo	1000	1000	892.6	224.9	1000	29.7	1000	1
Cultural	1000	1000	1000	1000	1000	76	1000	1000
Virulence	385.7	1000	714.3	192.7	83.1	25.2	56.1	721.1

Table 12
The required operational time (computational cost) for convergence after 100 iterations (millisecond).

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
GA	–	–	–	–	–	–	–	–	–	–	–
PSO	462	1365	543	897	2038	666	1531	978	812	877	1565
Cuckoo	14,199	16,980	15,819	16,420	18,193	15,975	16,041	16,009	15,741	7999	16,021
Cultural	917	1477	696	904	1366	975	1011	1044	777	903	937
Virulence	910	1320	683	888	2147	883	962	973	819	719	819

Table 13
the Parameter settings of the well-known optimization algorithms in the second test.

COA

Parameter	Population	Max. eggs	Min. eggs	Clusters	No. of variables	Boundary
Value	1000	10	2	5	100	[−25,25]

PSO

Parameter	Population	Social constant	Cognitive constant	No. of variables	Boundary
Value	1000	2	2	100	[−25,25]

Cultural

Parameter	Population	Alpha	Beta	Acceptance rate	No. of variables	Boundary
Value	1000	0.25	0.5	0.35	100	[−25,25]

Genetic algorithm

Parameter	Population	Crossover prob.	Mutation prob.	No. of variables	Boundary
Value	1000	0.75	0.1	100	[−25,25]

Virulence Optimization Algorithm

Parameter	Cell and virus population	Mutation (Drift) prob.	Reproduction (Shifting) Prob.	Clusters and clone no.	No. of variables	Boundary
Value	1000 and 100	0.1	0.75	5 and 10	100	[−25,25]

the test functions in complete 100 iterations so we can calculate the operational time of each algorithm on all the test functions. Table 12 illustrates the results. All the results are in millisecond (ms). It should be noted that the results are obtained on a computer with a Core2Duo 2.0 GHz CPU, 2GB RAM and 500GB HDD.

As it is seen from Table 8. The proposed algorithm outperforms the Cuckoo algorithm. It also performs relatively similar to PSO and cultural algorithm. Furthermore, we can conclude that the Virulence Optimization Algorithm is efficient in terms of operational time and computational cost.

4.2.2. Evaluation results of the second test

Table 13 illustrates the parameter settings of the well-known optimization algorithms used for comparison in the second test.

To evaluate the ability of the proposed algorithm to handle high-dimensional optimization problems compared to other well-known, we test the virulence algorithm and the other four algorithms using all the test objective functions. The corresponding parameter setting of each algorithm is depicted in Table 13. Table 14 illustrates the minimization result of virulence, GA, Cuckoo, PSO and cultural algorithm on a random run (the random run for each algorithm is the run that achieves the best result among a set of five random runs). In these runs the permissible number of iterations is set to 1000. Table 15 illustrates the mean stopping iterations for convergence in 10 test runs for the aforementioned function using all the algorithms.

We now investigate the performance of optimization algorithms on each aforementioned test function.

1st function (F1). As it is seen from above tables, the best result to optimize this function is achieved using the proposed algorithm. Only the proposed algorithm has been able to converge to the global optimum solution.

2nd function (F2). It is the only function that this proposed algorithm could not perform better than or similar to the others. The best result for optimization of this function is achieved using the proposed algorithm and the cuckoo optimization algorithm. The interesting point is that, contrary to the previous test, the PSO algorithm has not been able to converge to the optimal solution which indicates the inability of the PSO algorithm to handle the 2nd function in high-dimensional search space. The same can be said for GA and cultural.

3rd function (F3). Only the proposed algorithm and the Cuckoo algorithm have been able to converge to the global optimum solution, yet the proposed algorithm reached the convergence in fewer iterations than Cuckoo algorithm. Also, the cultural algorithm gets

Table 14
Convergence value and the iterations needed to converge after 1000 iterations.

Virulence	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Convergence value	0.0	8.612	0.0	0.0	4.9504	0.0	0.0	4.44E−16	−0.247202	−4.49E+307	0.0015684
Iterations	390	1000	720	200	1000	34	35	1000	12	1000	11
PSO	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Convergence value	3.6488	654.396	14.1364	10.4482	12.6705	278.1006	0.0	2.0141	1.33E−40	−1.323E+79	0.0015669
Iterations	1000	1000	1000	1000	1000	1000	44	1000	1000	1000	16
COA	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Convergence value	1015.27	17.88	0.0	0.0	6.9033	0.0	0.0	8.8818E−16	4.035E−34	−51.529	0.001568
Iterations	1000	1000	833	231	1000	33	16	1000	1000	1000	51
Cultural	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Convergence value	10569.033	2.76E+08	828.5752	1.40E+08	0.9998	12019.5	1.67E−05	19.1595	7.28E−06	−4.54E+106	0.0015687
Iterations	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	492
GA	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Convergence value	0.105706	126.0196	3.047033	0.06986	11.7187	18.02939	1.46E−13	0.060905	0.001866	−34.2605	0.001567
Iterations	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	103

Table 15
the mean stopping iterations required for convergence in 10 test runs.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
GA	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	106.6
PSO	1000	1000	1000	1000	1000	1000	45.8	1000	1000	1000	19.8
Cuckoo	1000	1000	843.1	238.5	1000	36.4	18.9	1000	1000	1000	60.5
Cultural	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	714.7
Virulence	390.3	1000	720.9	200.5	1000	35	35.6	1000	28.3	1000	11

stuck in the first local optima it finds and does not extend well to high-dimensional problems

4th function (F4). Again, only the proposed algorithm and the Cuckoo algorithm have been able to converge to the global optimum solution, yet the proposed algorithm reached the convergence in fewer iterations than Cuckoo algorithm. The interesting point about this result is that the GA score better compared to the PSO in terms of convergence value. Also, the cultural algorithm gets stuck in the first local optima it finds and does not extend well to high-dimensional problems

5th function (F5). Contrary to the previous results, the most robust performance on this function is achieved using cultural algorithm and the second best performance belongs to the proposed algorithm.

6th function (F6). As it is seen from above tables, the best result to optimize this function is achieved using the proposed algorithm and COA. Only the proposed algorithm and the COA algorithm have been able to converge to the global optimum solution. The results are nearly identical for these two algorithms. On the other hand, the other optimization algorithm has converged to the local minimum solutions.

7th function (F7). Only the proposed algorithm, PSO and COA have been able to reach the global optimum solution of zero. Among them, the Cuckoo converges to optimum solution in fewer iterations and the proposed algorithms has the second best performance. Other algorithms have converged to the local minimum solutions.

8th function (F8). The results obtained by the proposed algorithm and COA algorithm are fairly similar. However, the proposed algorithm has converged to a better minimum value compared to the other algorithm. On the other hand, all the other algorithms have converged to the local minimum solutions.

9th function (F9). The only algorithm that has been able to reach the global optimum solution is the proposed algorithm. The other algorithms have failed in reaching the global optimum solution.

10th function (F10). In this function the proposed algorithm has converged to a better minimum value compared to the other algorithms.

11th function (F11). The results obtained by all the algorithms are similar. However, the proposed algorithm has converged to the optimal solution in fewer iterations compared to the other algorithms.

According to the illustrated results, several interesting points can be perceived: (1) regarding the cultural algorithm, it is obvious, the algorithm demonstrates very poor results in high-dimensional search space and gets stuck in the first local minimum solution that it finds. (2) Contrary to the first round of test, the PSO algorithm exhibits worse than expected results. In other words, the algorithm is not equipped to handle high-dimensional optimization problems. (3) In general, the proposed algorithm outperforms the other algorithm in most of the times, except for function 7th and 5th. The COA algorithm comes in the second place.

According to the results depicted in the last two tables, an important matter can be perceived. The proposed algorithm not only outperform other algorithms in terms of convergence speed and the ability to converge to the optimal solution (in general) but also extends well to high-dimensional optimization problem and achieves good results in high-dimensional search space.

In the last stage of evaluation, the time of convergence for each algorithm after 100 iterations is calculated and the results are compared. In other words, all the algorithms are evaluated on all the test functions in complete 100 iterations so we can calculate the operational time of each algorithm on all the test functions. Table 16 illustrates the results. All the results are in second.

As it is obvious, the results illustrated in Table 16 demonstrate very good operational time for PSO algorithm. However, bear it in mind that the PSO algorithms exhibited worse than expected results in conducted experiments and these results cannot be considered as its strength point since it performed very poorly in other experiments. On the other hand, compared to other optimization

Table 16
The required operational time (computational cost) for convergence after 1000 iterations (seconds).

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
GA	–	–	–	–	–	–	–	–	–	–	–
PSO	6.395	20.599	7.461	13.191	19.22	9.5	18.65	13.99	9.56	25.2	16.51
Cuckoo	315.85	413.51	312.77	363.55	333.58	36.545	45.21	336.01	316	308.76	311.28
Cultural	389.37	816.63	712.5	817.14	744.05	698.24	1184.8	627.35	7167.1	148.99	641.25
Virulence	219.73	389.27	304.63	115.2	105.39	182.35	211.24	325.45	267.12	179.15	119.17

Table 17
The parameter settings of the designed test to assess the effectiveness of the proposed algorithm.

Parameters	Parameter value
Cell population	100
Virus population	10
Mutation (Drift) prob.	0.1
Reproduction (Shifting) prob.	0.75
Clusters no.	3
Clone no.	10
No. of variables	10
Lower bound.	−25
Upper bound.	25

algorithms, the proposed algorithm either performed better or relatively similar to them. Furthermore, we can conclude that the Virulence Optimization Algorithm is efficient in terms of operational time and computational cost in high-dimensional search spaces.

4.2.3. Assessing the effectiveness of the proposed virulence algorithm in achieving the global optimum solution

In order to assess the effectiveness of the proposed virulence algorithm in obtaining the optimal solution, all the test objective functions, excluding the 10th function (because the standard deviation of this function is found to be *Inf*), are used. Measuring the effectiveness of the proposed algorithm is conducted through hypothesis testing. At first, the effectiveness of the proposed algorithm in reaching the global optimum solution of the 1st test objective function is evaluated. To do so, 100 trial runs of the algorithm with 100 iterations, each are conducted using the 1st function. The normal distribution is assumed for this experiment. The parameter settings of this experiment are illustrated in Table 17.

It should be noted that the global optimum value of the first function is 0.0 (zero). Accordingly, the mean and sample standard deviation of the recorded optimum values of the trial runs is found to be 3.1040e−81 and 5.9797e−81, respectively. Let the null hypothesis be as follows (it should be noted that at the 0.01 level of significance, the effectiveness of the proposed algorithm on the 1st test objective function needs to achieve at least 0.99):

$$H_0 : \bar{f} \geq 0.99, \qquad H_A : \bar{f} < 0.99, \tag{18}$$

In this formula, $\bar{f}$ represents the mean of the sample. In order to assess whether the null hypothesis should be accepted or rejected, first we need to calculate the observed z value as follows:

$$z = \frac{mean\text{-}effectiveness \quad value}{standard \quad deviation/\sqrt{number \quad of \quad samples}} = \frac{3.1040e^{-81} - 0.99}{5.9797e^{-81}/\sqrt{100}} = 1.6556e^{+81} \tag{19}$$

Since $z_{0.01} = 2.33 < 1.6556e^{+81}$, the null hypothesis is accepted.

Generally, consider a null hypothesis of $H_0 : \bar{f} \geq 1 - t$, t is a non-negative value. Suppose the considered hypothesis is accepted at the 0.01 level of significance. This insinuates that the normalized observed mean value needs to be at least 2.33.

$$\frac{3.1040e^{-81} - (1-t)}{5.9797e^{-81}/\sqrt{100}} = 2.33 \rightarrow t = 1 \rightarrow (1-t) = 0 \tag{20}$$

As a result, the null hypothesis of $H_0 : \bar{f} \geq 0$ is acceptable at 0.01 level of significance.

The result of the hypothesis testing on the rest of the test objective functions is illustrated in Table 18.

The results illustrated in Table 18 suggests the proposal virulence algorithm not only performs well in terms of the convergence speed and the required operational time, but also is effective and reliable in finding the optimum values of the test objective functions.

5. Discussion

In the previous section, we evaluated the proposed virulence algorithm in terms of convergence capability, convergence speed and the operational time required for the convergence. As it is evident from the results depicted in the previous section, while most optimization algorithms could not reach the global optimum solution, virulence algorithm has been able to reach the global solution of test functions or reach a much better cost values than others and therefore has a better convergence capability than the others. Of course the Rosenbrock test function is the exception where the PSO algorithm achieves better results than the proposed algorithm. In terms of convergence speed, although some of the optimization algorithms have converged to similar results, the proposed algorithm has outperformed all the algorithms because it has

Table 18
The result of the hypothesis testing on the test objective functions.

	Mean	Standard deviation	Global optimum	z-Score	t	$1-t$	Null hypothesis
1st function	3.1040e−81	5.9797e−81	0.0	1.6556e+81	1	0.0	Accepted
2nd function	0.0089	1.5324e−04	0.0	6.4025e+04	0.9911	0.0089	Accepted
3rd function	6.9030e−41	3.6464e−41	0.0	2.7150e+41	1	0.0	Accepted
4th function	0.0	0.0	0.0	Inf	1	0.0	Accepted
5th function	0.1115	0.0218	0.0	402.8869	0.8936	0.1064	Accepted
6th function	0.0	0.0	0.0	Inf	1	0.0	Accepted
7th function	0.0	0.0	0.0	Inf	1	0.0	Accepted
8th function	5.1514e−16	4.9989e−16	0.0	1.9805e+16	∼1	4.4409e−16	Accepted
9th function	−0.2454	0.0069	−0.247	1.7792e+03	1.2470	−0.2470	Accepted
11th function	0.0016	3.3143e−04	0.0015685	2.9822e+04	0.9985	0.0015	Accepted

Level of significance = 0.01, $z_{0.01} = 2.33$.

been able to reach the optimal solution in fewer iterations than the other algorithms. From the results depicted in the last section, it can be perceived that the proposed algorithm extends well to high-dimensional search spaces. The escape mechanism plays an important role in convergence speed, because every time the viruses intend to escape to a region with maximum resources, the search space is clustered into several regions and the region with the highest probability of containing optimal solution is selected as escape destination. During this process, the search space continues to shrink until the majority of the viruses converges to the optimal solution. Therefore the escape mechanism facilitates the process of searching for an optimal solution. In terms of the required operational time for convergence, the proposed algorithm has proved to be efficient. It has either performed relatively similar to PSO and Cultural or outperformed other algorithms. The best performance of the proposed algorithm is achieved by evaluating the algorithm on fifth De Jong function. This function has a lot of optimal solutions and most optimization algorithms get stuck in the first peak the find. As it is obvious from the results all the other algorithms get stuck in a local minimum solution. However, the proposed algorithm successfully reached the global minimum solution. Through this observation, it can be concluded that the proposed algorithm can escape local minimum where other optimization algorithms get stuck.

6. Conclusion

In this paper, a new optimization algorithm to solve optimization problems has been introduced which is inspired by the optimal mechanism of viruses when infecting body cells. Special mechanism and function of viruses which includes the recognition of fittest viruses to infect body cells, reproduction (cloning) of these cells to prompt "invasion" operation of ready-to-infect regions and then escaping from infected regions (to avoid immune reaction) is the basis of this evolutionary optimization algorithm. There are two types of population in this algorithm. First, the host cell population, which represents the resources available in host environment or the region containing the global optimum solution. Second, the virus population, which infiltrates the host environment and attempts to infect it. In the optimization procedure, at first the viruses reside in the constituted regions or clusters of the environment called virus groups (via K-means clustering). Then they scatter in host environment through mutation (Drifting) and recombination (Shifting) operators and search the host environment for a region which possesses the maximum vital resources for survival and reproduction. Then among the virus groups, the group with highest mean fitness is chosen as escape destination and the virus with the highest fitness in this group is chosen as the escape point which all the mature viruses escape to. The selected group or cluster represents a region in host environment which most likely possesses the maximum vital resource. Before the escape operation commences, best viruses in each virus group are recognized and undergoes a cloning operation. This operation results in a higher level of virulence. At the end of each generation, the viruses with worst fitness are recognized and get eliminated due to immune response so that the equilibrium of environment is maintained. This procedure continues until the majority of virus population is gathered in the region containing maximum resources or the global optimum solution. The proposed virulence optimization algorithm has been evaluated using 11 test functions. Comparing the result of this algorithm with other well-known optimization algorithms such as GA, PSO, cuckoo optimization algorithm and cultural algorithm demonstrate the superiority of the proposed algorithm in the convergence to the global optimum solution and also the convergence speed. The result of evaluation of the 5th test function suggests the former (Tables 2–6). In most of the conducted experiments, the proposed algorithm converged to the global optimum solution in fewer iterations. Also in cases where none of the other algorithms have been able to converge to the global optimum solution (5th Function or F5), the proposed algorithm has been able to converge to the global optimum in the least iterations possible (82 iterations). Also, it can be concluded that the proposed algorithm extends well to High-dimensional optimization problems. At last, the proposed algorithm proves to be effective in finding the optimum solution of the test objective functions. It should be noted that the higher performance of the virulence algorithm in converging to the global optimum solution or obtaining better optimization results on test functions does not necessarily mean that virulence algorithm is the best evolutionary algorithm ever developed. It just can be considered as a successful emulation of evolutionary behaviours in nature; suitable for some optimization problems.

References

[1] J. Nocedal, S.J. Wright, Numerical Optimization Springer Series in Operations Research and Financial Engineering, 2006.
[2] R. Rajabioun, Cuckoo optimization algorithm, Appl. Soft Comput. 11 (8) (2011) 5508–5518.
[3] R.L. Haupt, S.E. Haupt, Practical Genetic Algorithms, second ed., John Wiley & Sons Inc., 2004.
[4] B. Korte, J. Vygen, Combinatorial optimization Theory and Algorithms Series: Algorithms and Combinatorics, vol. 21, 5th ed., 2012.
[5] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, A. Schrijver, Combinatorial Optimization, John Wiley & Sons Inc., 1998.
[6] R.E. Bellman, Dynamic Programming, Dover Publications, Incorporated, 2003.
[7] S.R. Eddy, What is dynamic programming? Nat. Biotechnol. 22 (July (7)) (2004) 909–910.
[8] J.A. Snyman, Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms, Springer Publishing, 2005.
[9] J.C. Spall, Introduction to Stochastic Search and Optimization, Wiley, 2003.
[10] J. Schneider, S. Kirkpatrick, Stochastic Optimization Scientific Computation Series, Springer, 2006.
[11] M. Melanie, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, USA, 1999.
[12] S.N. Sivanandam, S.N. Deepa, Introduction to Genetic Algorithms, Springer-Verlag, Berlin, Heidelberg, 2008.
[13] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, 1995, pp. 1942–1948.
[14] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, 1st ed., John Wiley & Sons Inc., 2005.
[15] M. Dorigo, C. Blum, Ant colony optimization theory: a survey, Theor. Comput. Sci. 344 (2–3) (2005) 243–278.
[16] M. Dorigo, L.M. Gambardella, Ant Colony System: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolut. Comput. 1 (1) (1997) 53–66.
[17] L.N. de Castro, J.I. Timmis, Artificial immune systems as a novel soft computing paradigm, Soft Comput. 7 (8) (2003) 526–544.
[18] D. Dasgupta, S. Yu, F. Nino, Recent advances in artificial immune systems: models and applications, Appl. Soft Comput. 11 (2011) 1547–1587.
[19] E. Atashpaz-Gargari, C. Lucas, Imperialist Competitive Algorithm: an algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4666.
[20] R.L. Johnston, H.M. Cartwright, Application of Evolutionary Computing in Chemistry, Springer-Verlag, Berlin Heidelberg, 2004.
[21] V. Cheldaboina, M.K. Ranga, Reduced order optimal control using genetic algorithms, in: American Control Conference, Portland, USA, 2005.
[22] W. Zhao, C.E. Davis, A modified artificial immune system based pattern recognition approach—an application to clinical diagnostics, Artif. Intell. Med. 52 (1) (2011) 1–9.
[23] L. Hong, A novel particle swarm optimization method using clonal selection algorithm, in: International Conference on Measuring Technology and Mechatronics Automation, vol. 2, 2009, pp. 471–474.
[24] T.C. Barrett, P. Pastoret, W.J. Taylor, Virus Plagues of Large and Small Ruminants, Elsevier Academic Press, Amsterdam, 2006.
[25] T.T. Lam, C.C. Hon, J.W. Tang, Use of phylogenetics in the molecular epidemiology and evolutionary studies of viral infections, Crit. Rev. Clin. Lab. Sci. 47 (1) (2010) 5–49.
[26] E. Domingo, C. Escarmís, N. Sevilla, A. Moya, S.F. Elena, J. Quer, I.S. Novella, J.J. Holland, Basic concepts in RNA virus evolution, FASEB J. 10 (8) (1996) 859–864.
[27] C.L. Boutwell, M.M. Rolland, J.T. Herbeck, J.I. Mullins, T.M. Allen, Viral evolution and escape during acute HIV-1 infection, J. Infect. Dis. (2010).
[28] J. Chen, Y.M. Deng, Influenza virus antigenic variation, host antibody production and new approach to control epidemics, Virol. J. 6 (2009).

[29] S. Alizon, A. Hurford, N. Mideo, M. Van Baalen, Virulence evolution and the trade-off hypothesis: history, current state of affairs and the future, J. Evolut. Biol. 22 (2) (2009) 245–259.
[30] S. Astier, J. Albouy, V. Moury, C. Robaglia, H. Lecoq, Principles of Plant Virology, Genome, Pathogenicity, Virus Ecology, 1st ed., Science Pub Inc., 2008.
[31] E. Domingo, C.R. Parrish, J.J. Holland, Origin and Evolution of Viruses, Academic Press, 2008.
[32] K. Leppard, N. Dimmock, A. Easton, Introduction to Modern Virology, Blackwell Publishing Limited, 2007.
[33] W.J. Mahy, M.H.V. Van Regenmortel, Desk Encyclopedia of General Virology, Oxford Academic Press, 2009.
[34] J. Bertram, The molecular biology of cancer, J. Mol. Asp. Med. 21 (6) (2000) 167–223.
[35] V. Burrus, M. Waldor, Shaping bacterial genomes with integrative and conjugative elements, Res. Microbiol. 155 (5) (2004) 376–386.
[36] P.J. Hastings, J.R. Lupski, S.M. Rosenberg, G. Ira, Mechanisms of change in gene copy number, J. Nat. Rev. Genet. 10 (8) (2009) 551–564.
[37] B.A. Montelone, Mutation, Mutagens, and DNA Repair, Division of Biology, Kansas State University, 1998.
[38] P.R. Murray, K.S. Rosenthal, M.A. Pfaller, Medical Microbiology, 6th ed., Saunders Pub., 2012.
[39] M. Molga, C. Smutnicki, Test Functions for Optimization Needs, 2005, Retrieved from: http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf (June 2014).
[40] A. Chipperfield, P.J. Fleming, H. Pohlheim, C.M. Fonseca, Genetic Algorithm Toolbox for use with Matlab, Technical Report No. 512, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994.
[41] A.J. Chipperfield, P.J. Fleming, H. Pohlheim, A genetic algorithm toolbox for MATLAB, in: Proceedings of Int. Conf. On Sys. Engineering, Coventry, UK, 1994, pp. 200–207.
[42] L.A. Rastrigin, Systems of extremal control, in: Theoretical Foundations of Engineering Cybernetics Series, Nauka, Russian, Moscow, 1974.
[43] H. Mühlenbein, D. Schomisch, J. Born, The parallel genetic algorithm as function optimizer, Parallel Comput. 17 (1991) 619–632.
[44] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm Evolut. Comput. 1 (March (1)) (2011) 32–49.
[45] R. Storn, K. Price, Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Tech. Rep. TR-95-012, ICSI, 1995.
[46] K.V. Price, An introduction to differential evolution, in: New Ideas in Optimization, McGraw-Hill, 1999, pp. 79–108.
[47] R. Sarker, H. Abbass, Differential evolution for solving multi-objective optimization problems, Asia Pacific J. Oper. Res. 21 (2) (2004) 225–240.
[48] W. Gong, Z. Cai, An improved multiobjective differential evolution based on Pareto-adaptive epsilon-dominance and orthogonal design, Eur. J. Oper. Res. 198 (2) (2009) 576–601.
[49] M. Gong, L. Jiao, H. Du, L. Bo, Multiobjective immune algorithm with nondominated neighbor-based selection, Evolut. Comput. 16 (2) (2008) 225–255.
[50] J. Chen, Q. Lin, Z. Ji, A hybrid immune multiobjective optimization algorithm, Eur. J. Oper. Res. 204 (2) (2010) 294–302.
[51] Z.-H. Hu, A multiobjective immune algorithm based on a multiple-affinity model, Eur. J. Oper. Res. 202 (1) (2010) 60–72.
[52] J. Moore, R. Chapman, Application of Particle Swarm to Multiobjective Optimization, Tech. Rep., Department of Computer Science and Software Engineering, Auburn University, 1999.
[53] A. Elhossini, S. Areibi, R. Dony, Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization, Evolut. Comput. 18 (1) (2010) 127–156.
[54] S. Agrawal, B.K. Panigrahi, M.K. Tiwari, Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch, IEEE Trans. Evolut. Comput. 12 (5) (2008) 529–541.
[55] V.L. Huang, P.N. Suganthan, J.J. Liang, Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems, Int. J. Intell. Syst. 21 (2) (2006) 209–226.
[56] R.Y. Rubinstein, D.P. Kroese, The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning, Springer Verlag, 2004.
[57] K.-H. Han, J.-H. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, in: IEEE Congress on Evolutionary Computation, CEC 2000, 2000, pp. 1354–1360.
[58] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, IEEE Trans. Evolut. Comput. 6 (2002) 580–593.
[59] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions I: binary parameters, in: Parallel Problem Solving from Nature, PPSN IV, LNCS, vol. 1411, 1996, pp. 178–187.
[60] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[61] K.I. Smith, R.M. Everson, J.E. Fieldsend, C. Murphy, R. Misra, Dominance-based multiobjective simulated annealing, IEEE Trans. Evolut. Comput. 12 (3) (2008) 323–342.
[62] Leonora FF B., M. Dorigo, L.M. Gambardella, W.J. Gutjahr, A survey on metaheuristics for stochastic combinatorial optimization, Nat. Comput. 8 (2) (2009) 239–287.
[63] R. Caballero, M. González, F.M. Guerrero, J.M. Luque, C. Paralera, Solving a multiobjective location routing problem with a metaheuristic based on Tabu search. Application to a real case in Andalusia, Eur. J. Oper. Res. 177 (3) (2007) 1751–1763.
[64] L. Belfares, W. Klibi, N. Lo, A. Guitouni, Multi-objectives Tabu search based algorithm for progressive resource allocation, Eur. J. Oper. Res. 177 (3) (2007) 1779–1799.
[65] R.P. Beausoleil, MOSS: multiobjective scatter search applied to non-linear multiple criteria optimization, Eur. J. Oper. Res. 169 (2) (2006) 426–449.
[66] A.P. Reynolds, B. de la Iglesia, A multi-objective GRASP for partial classification, Soft Comput. 13 (3) (2009) 227–243.
[67] L. Astudillo, P. Melin, O. Castillo, Introduction to an optimization algorithm based on the chemical reactions, Inf. Sci. (2014).
[68] W.W.P. Tsang, H.Y.K. Lau, An artificial immune system-based many-objective optimization algorithm with network activation scheme, Adv. Artif. Life ECAL (2013).
[69] O. Castillo, H. Neyoy, J. Soria, P. Melin, F. Valdez, A new approach for dynamic fuzzy logic parameter tuning in Ant Colony Optimization and its application in fuzzy control of a mobile robot, Appl. Soft Comput. 28 (March) (2015) 150–159.
[70] A. Rahimi-Vahed, S.M. Mirghorbani, M. Rabbani, A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem, Soft Comput. 11 (10) (2007) 997–1012.
[71] B. Alatas, E. Akin, A. Karci, MODENAR: multi-objective differential evolution algorithm for mining numeric association rules, Appl. Soft Comput. 8 (1) (2008) 646–656.
[72] L. Sánchez, J. Otero, I. Couso, Obtaining linguistic fuzzy rule-based regression models from imprecise data with multi-objective genetic algorithms, Soft Comput. 13 (5) (2009) 467–479.
[73] S.-Y. Shin, I.-H. Lee, D. Kim, B.-T. Zhang, Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing, IEEE Trans. Evolut. Comput. 9 (2) (2005) 143–158.
[74] E. Aggelogiannaki, H. Sarimveis, A simulated annealing algorithm for prioritized multiobjective optimization-implementation in an adaptive model predictive control configuration, IEEE Trans. Syst. Man Cybern. B: Cybern. 37 (4) (2007) 902–915.
[75] B. Lazzerini, F. Marcelloni, M. Vecchio, A multi-objective evolutionary approach to image quality/compression trade-off in JPEG baseline algorithm, Appl. Soft Comput. 10 (2) (2010) 548–561.
[76] J. Handl, J.D. Knowles, An evolutionary approach to multiobjective clustering, IEEE Trans. Evolut. Comput. 11 (1) (2007) 56–76.
[77] I. Aydin, M. Karaköse, E. Akin, A multi-objective artificial immune algorithm for parameter optimization in support vector machine, Appl. Soft Comput. 11 (1) (2011) 120–129.
[78] A. Asllani, A. Lari, Using genetic algorithm for dynamic and multiple criteria web-site optimizations, Eur. J. Oper. Res. 176 (3) (2007) 1767–1777.