Engineering Optimization, 2015
http://dx.doi.org/10.1080/0305215X.2014.994868

A novel metaheuristic for continuous optimization problems: Virus optimization algorithm

Yun-Chia Liang[a,b]* and Josue Rodolfo Cuevas Juarez[a]

[a]*Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan, Taiwan;*
[b]*Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Taoyuan, Taiwan*

(*Received 11 December 2013; accepted 11 November 2014*)

A novel metaheuristic for continuous optimization problems, named the virus optimization algorithm (VOA), is introduced and investigated. VOA is an iteratively population-based method that imitates the behaviour of viruses attacking a living cell. The number of viruses grows at each replication and is controlled by an immune system (a so-called 'antivirus') to prevent the explosive growth of the virus population. The viruses are divided into two classes (strong and common) to balance the exploitation and exploration effects. The performance of the VOA is validated through a set of eight benchmark functions, which are also subject to rotation and shifting effects to test its robustness. Extensive comparisons were conducted with over 40 well-known metaheuristic algorithms and their variations, such as artificial bee colony, artificial immune system, differential evolution, evolutionary programming, evolutionary strategy, genetic algorithm, harmony search, invasive weed optimization, memetic algorithm, particle swarm optimization and simulated annealing. The results showed that the VOA is a viable solution for continuous optimization.

Keywords: virus optimization algorithm; metaheuristic; continuous optimization; global optimization

1. Introduction

A metaheuristic can be thought of as a set of computer-based instructions that allow computers to simulate real-world phenomena. This adaptation of natural phenomena in computers has helped in addressing problems that were previously difficult or impossible to solve. To overcome several deficiencies in conventional optimization methods, extensive stimulations have been developed to investigate social behaviour, *e.g.* particle swarm optimization (PSO) (Kennedy and Eberhart 1995), ant colony optimization (ACO) (Dorigo 1992), artificial bee colony (ABC) (Pham *et al.* 2005), bacterial foraging optimization (Liu and Passino 2002) and harmony search (HS) (Lee and Geem 2005); physical phenomena, *e.g.* simulated annealing (SA) (Kirkpatrick, Gelatt, and Vecchi 1983) and quantum-inspired metaheuristics (Balicki 2009; Li, Song, and Yang 2010); and biological systems, *e.g.* artificial immune systems (AIS) (Farmer, Packard, and Perelson 1986) and genetic algorithms (GAs) (Holland 1975). Metaheuristic algorithms are indeed powerful tools in today's optimization field; however, they are not guaranteed to converge to the global optimum in each instance. Therefore, this study presents a novel approach which can be easily implemented in the computational realm to create ever better and simpler algorithmic tools.

*Corresponding author. Email: ycliang@saturn.yzu.edu.tw

The newly developed metaheuristic, named the virus optimization algorithm (VOA) (Cuevas *et al.* 2009), imitates the behaviour of a virus attacking a host cell. During the attack, the virus tries to efficiently exploit the cell's resources (also known as the cell's machinery) by creating more viruses. When it locates the cell's nucleolus, the virus causes the death of the host cell. VOA mainly focuses on converting the concept of a virus attacking a host cell into a continuous domain optimization method. VOA's small set of controllable parameters makes it easy to implement and it can quickly produce a reasonable solution for problems too difficult to be solved by conventional methods.

The authors first presented VOA as a proceedings work (Cuevas *et al.* 2009) but the current version differs in several important ways. First, this article introduces a very comprehensive background and analogy of VOA. Secondly, the number of parameters in VOA implementation has been reduced from 12 to 5 without a reduction in terms of efficiency or effectiveness. Thirdly, the robustness of VOA has been extensively tested in highly complex problems with dimensionality ranging up to 1000 variables. The benchmark functions with shifted and rotated factors were also considered. Lastly, the performance of VOA is validated more thoroughly here by comparing the results with over 40 state-of-the-art optimization techniques from the literature.

The remainder of this study is organized as follows. Section 2 introduces the formal terminology used for the algorithm. Section 3 gives a detailed explanation of VOA. Section 4 discusses the computational results using the proposed algorithm, where several continuous benchmark functions and well-known metaheuristics from the literature are used for comparison. Section 5 concludes this study and addresses future research directions.

2. Background

This section provides necessary background for the proposed VOA such as the origin and basic information of the virus, interaction between virus and cell, and cell self-defence mechanism (*i.e.* immune system), and also explains the formal terminology used throughout this study.

There are three main theories of the origin of viruses (Russell 2001; Shors 2009; Topley *et al.* 1998), specifically that viruses may have evolved from plasmids, from bacteria, or from complex molecules of protein and nucleic acid. As infectious agents, viruses consist of a genetic element, a protein coat and an envelope (*i.e.* bacteriophage). To multiply or replicate, viruses need to enter cells through infection. The infected cell is called the host (Madigan, Martinko, and Parker 2006). Viruses exploit the cell's metabolic mechanism and alter its production of protein. This alteration in the cell metabolism can create several new viruses (Flint *et al.* 1999; Madigan, Martinko, and Parker 2006). The virus reproduction (replication) inside the host eventually causes the death of the cell.

During the virus replication process, the host cell can be affected at the macromolecular level owing to infection. These cytopathic effects, or degenerative changes in cells, can be extensive (Riss and Moravec 2004; Roulston, Marcellus, and Branton 1999; Topley *et al.* 1998). Cell death starts from the outer layer or membrane and then extends to the cell's nucleolus or nucleoid. The virus replication rate varies widely among different virus–host combinations. However, viruses with strong DNA or RNA structures (*i.e.* less common viruses) are more likely to reproduce at a higher rate in which they simultaneously attach to the same host cell, a process also known as coinfection (Flint *et al.* 1999).

Immunity to viruses is developed over time, with B lymphocytes (also known as B cells) playing an essential role (Davidson 2010; Salyers and Whitt 2001). Stimulated B cells produce many special molecules called antibodies (Tanegawa 1983), which are highly selective and attack only one type of virus. When the viruses replicate and start to interact with the host cell, the defence mechanism (immune system) is alerted, triggering the production of antibodies in an attempt to suppress the infection and prevent the death of the host cell. Despite the immune system's efforts to protect the host cell, no defence mechanism offers complete protection. Therefore, slowing down infections of viruses with a high mutation probability, such as RNA viruses, present a major challenge to this defence mechanism during the attack (Campbell and Reece 2001).

3. The virus optimization algorithm

3.1. *Analogy*

Based on the background provided in Section 2, several analogies between the behaviour of a virus attacking a cell and the proposed VOA are introduced, as follows. First, in VOA the cell itself will be considered as the solution space or feasible space for the problem. Secondly, the global optimum is located inside the solution space and, by analogy, that location will be represented by the nucleolus, which synthesizes the proteins needed for cell replication. Thirdly, the VOA user has to determine the number of viruses that can coexist harmlessly in the host cell. The location of each virus inside (and only inside) the host cell will represent one complete solution; that is, if the target is an n-dimensional continuous optimization problem, the value for each variable in each dimension will determine the location of the virus.

The replication of the viruses is the fourth analogy. The algorithm will trigger interaction between those members that harmlessly coexist with the host cell. In other words, these viruses will start to create new viruses. In this analogy, two types of virus will be considered: strong viruses that have good structures (in terms of optimization, 'good structures' are defined by the objective function value) and common viruses with insufficient structure (or strength). The common viruses are all the members in the population of viruses without considering the strong members. In this analogy, strong viruses are replicated at a higher rate because strong members exploit the machinery of the cell more efficiently, as noted in Section 2.

In the fourth analogy, the common members are dedicated to exploring the cell (solution space), and the strong members focus on killing the cell (*i.e.* exploiting the areas where the global optima or cell nucleolus may be found). A small random perturbation is responsible for creating new members close to the strong viruses and far from the common ones. Therefore, this perturbation will differ for strong and common members, as detailed in Section 3.3.

The fifth and final analogy is the organism's immune system, which creates antibodies to stop virus replication. In the VOA, this system is referred to as the 'antivirus mechanism' and imitates the B lymphocytes, *i.e.* it fights against viruses that attack the host cell. Having stopped the replication procedure, the antivirus then exterminates some members of the virus population. Common viruses will be more likely to be stopped because of their weaker genetic structures (determined by the objective function value in VOA). However, replication is followed by the creation of new viruses, some of which may be stronger than their predecessors.

Unless the replication process produces stronger members (*i.e.* better solutions are generated) or cell death is achieved (*i.e.* the global optimum is found), after several replications the antivirus is more likely to exterminate the viruses. However, the portion of members which are considered as strong viruses will not be reduced, *i.e.* the antivirus does not kill any strong members unless they have been removed from the list of strong viruses and are categorized as common members.

The overall population of viruses consists of the newly created members and viruses from previous generations. This mechanism simulates the behaviour of the real immune system when protecting the host cell. The predecessors and new viruses will compete to survive in each replication through killing a given number of viruses according to Equation (1). Based on this equation, the number of members killed may change between replications.

$$\text{amount} = \text{rand}(0, \text{population\_size} - \text{strong\_members}) \qquad (1)$$

The population_size is the total number of viruses currently existing inside the host cell. The strong_members is a parameter set by the user, which is detailed in Section 3.2. Once the number of viruses to be killed is determined, those viruses with weak structures (*i.e.* inferior objective function values) are more likely to be killed, while members in the population with stronger structures (*i.e.* superior objective function values) will have a greater chance of survival.

3.2. *Parameters*

Once the behaviour of virus infection and organism immune system is understood, the controllable parameters of the proposed VOA are defined as follows:

1. Number of initial solutions: the size of the initial virus population coexisting harmlessly in the cell.
2. Number of strong viruses: the proportion of top solution(s) in the population.
3. The growth rate for generating new solutions (viruses) during the replication from the strong viruses: this rate determines the intensity of exploitation since the new solutions will be near to the current strong viruses.
4. The growth rate for generating new viruses (solutions) from the common members: this value controls the exploration strength because the new members will be created far from the current common viruses in the cell (solution space).
5. The stopping criterion: the number of consecutive non-improving solutions, maximum number of replications or discovery of the global optimum, *etc*.

3.3. *Procedure of the virus optimization algorithm*

The process in which a virus attacks a host cell during infection is brought into the realm of optimization through the development of a novel and fairly powerful algorithm. The proposed algorithm is a powerful population-based method that begins the search with a small number of viruses (solutions).

To simulate the replication process when new viruses are created, the *population size* between replications is dynamic, and the amount of viruses is controlled by the *antivirus* mechanism, which is responsible for protecting the host cell from virus attack. The whole process will be terminated based on the stopping criterion: the maximum number of iterations (*i.e.* virus replication) or the discovery of the global optimum (*i.e.* cell death is achieved).

The VOA consists of three main procedures: initialization, replication and updating/maintenance. The initialization procedure uses the values of each parameter (defined by the user) to create the first population of viruses. These viruses are ranked (sorted) based on the objective function evaluation to select strong and common members (following the concept introduced in the fourth analogy). Replication then produces new viruses (solutions) by considering the strong and common members. Maintenance/updating allows the algorithm to apply the antivirus mechanism (introduced in the fifth analogy) to control the number of members in the population. In addition, this process intensifies exploitation whenever the algorithm does not improve the average objective function values of the population following implementation of the antivirus mechanism. The pseudo-code of VOA is summarized in Figure 1, and the details are discussed in the following subsections.

```
VOA_metaheuristic **PSEUDO CODE**
Virus_population ← generate_initial_population(parameters_values); //Initialization
Viruses_strength ← Evaluate_objective_function_value(Virus_population);
While(TRUE)
    [Strong_viruses, Common_viruses ] ← Classification(Virus_population, Viruses_strength);
    [New_Strong, New_Common ] ← Replication(Strong_viruses, Common_viruses);//Replication
    New_members ← Storage(New_Strong, New_Common);
    New_members_strength ← Evaluate_objective_function_value(New_members);
    Virus_population ← Combine(Virus_population, New_members); //Updating
    Population_performance ← Average_objective_function_value(Virus_population);
    If (Population_performance did not improve) → Intensify_exploitation();
    Apply Antivirus();//Maintenance
    If (populationsize exceeds 1,000 members) → Reduction(Virus_population, Virus_strength, initial_value);
    Stop ← Evaluate_stopping_criterion();
    If (Stop == TRUE) → BREAK WHILE;
End while
```

Figure 1. Pseudo-code of the virus optimization algorithm (VOA).

3.3.1. *Initialization procedure*

As can been seen in many other metaheuristics, the initialization procedure in VOA consists of two parts: parameter setting and the generation of initial viruses. Parameter setting is a key performance aspect, and users commonly try to find appropriate ranges for the controllable parameters in their preliminary run tests. Since each parameter performs different tasks such as exploitation (strong viruses and their corresponding growth rate) and exploration (common viruses and their corresponding growth rate), the importance (order) of each parameter is not easy to determine. For this study, a three-level factorial design was used to set the VOA parameters, which is a widely used approach when interactions between parameters are considered.

After the parameter values have been set, the initial virus population is randomly generated (where the location of each virus represents a complete solution containing the values of each variable). Finally, once the initial viruses have been created, the replication counter is initialized to a value of 1.

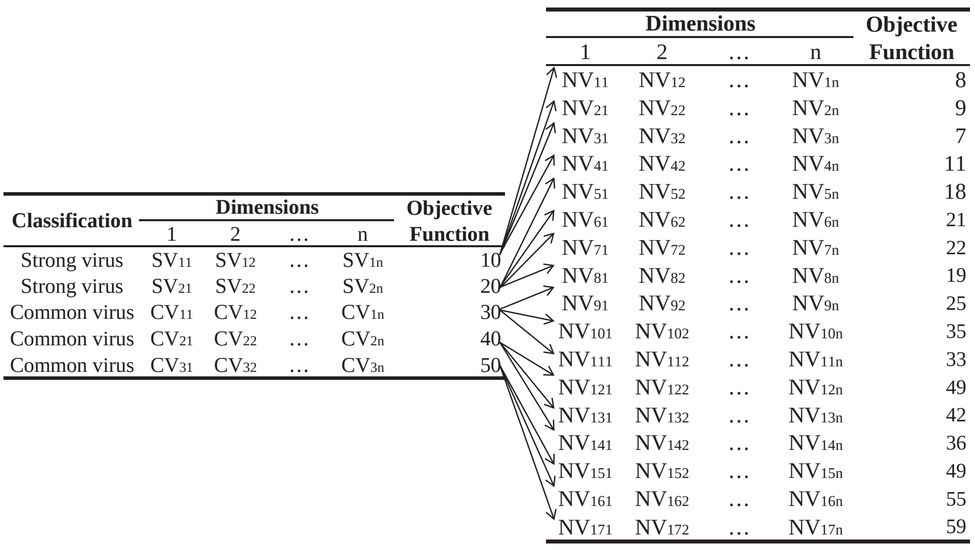
3.3.2. *Replication procedure*

3.3.2.1. *Classification of the viruses.* This step classifies the viruses into two levels: strong and common. As illustrated in Table 1, the initial population consists of five viruses ranked according to their objective function values (*i.e.* from the best to the worst). Assuming that the objective is to minimize some function and the number of strong viruses is set to two, the first two viruses in the list will then be considered strong while the remaining viruses are considered common.

3.3.2.2. *Replication of the viruses.* Here, the replication procedure is performed using the parameters defined by the user in the initialization stage described above, where a temporary matrix (larger than the matrix containing the original viruses) will hold the newly generated members. The procedure begins with the creation of four new viruses from the strong members and three from the common members (Figure 2). In this example, SV, CV and NV stand for strong, common and new viruses, respectively. VOA uses Equation (2) to generate new viruses from strong members considered, while Equation (3) generates new solutions (viruses) from

Table 1. Illustration of strong and common virus classification.

Classification	Rank	Dimensions 1	2	...	n	Objective function value
Strong virus	1	SV_{11}	SV_{12}	...	SV_{1n}	10
Strong virus	2	SV_{21}	SV_{22}	...	SV_{2n}	20
Common virus	3	CV_{11}	CV_{12}	...	CV_{1n}	30
Common virus	4	CV_{21}	CV_{22}	...	CV_{2n}	40
Common virus	5	CV_{31}	CV_{32}	...	CV_{3n}	50



Classification	Dimensions 1	2	...	n	Objective Function
Strong virus	SV_{11}	SV_{12}	...	SV_{1n}	10
Strong virus	SV_{21}	SV_{22}	...	SV_{2n}	20
Common virus	CV_{11}	CV_{12}	...	CV_{1n}	30
Common virus	CV_{21}	CV_{22}	...	CV_{2n}	40
Common virus	CV_{31}	CV_{32}	...	CV_{3n}	50

Dimensions 1	2	...	n	Objective Function
NV_{11}	NV_{12}	...	NV_{1n}	8
NV_{21}	NV_{22}	...	NV_{2n}	9
NV_{31}	NV_{32}	...	NV_{3n}	7
NV_{41}	NV_{42}	...	NV_{4n}	11
NV_{51}	NV_{52}	...	NV_{5n}	18
NV_{61}	NV_{62}	...	NV_{6n}	21
NV_{71}	NV_{72}	...	NV_{7n}	22
NV_{81}	NV_{82}	...	NV_{8n}	19
NV_{91}	NV_{92}	...	NV_{9n}	25
NV_{101}	NV_{102}	...	NV_{10n}	35
NV_{111}	NV_{112}	...	NV_{11n}	33
NV_{121}	NV_{122}	...	NV_{12n}	49
NV_{131}	NV_{132}	...	NV_{13n}	42
NV_{141}	NV_{142}	...	NV_{14n}	36
NV_{151}	NV_{152}	...	NV_{15n}	49
NV_{161}	NV_{162}	...	NV_{16n}	55
NV_{171}	NV_{172}	...	NV_{17n}	59

Figure 2. Example of a matrix for storing new viruses generated after performing the replication procedure.

common members considered.

$$\mathrm{NV}_{ij} = \mathrm{SV}_{ij} \pm \frac{\mathrm{rand()}}{\mathrm{intensity}} \times \mathrm{SV}_{ij} \tag{2}$$

$$\mathrm{NV}_{ij} = \mathrm{CV}_{ij} \pm \mathrm{rand()} \times \mathrm{CV}_{ij} \tag{3}$$

The subscripts *i* and *j* represent the *i*th member in the population on the *j*th dimension. Equations (2) and (3) are used to balance exploitation and exploration, since Equation (2) generates new members close to the strong viruses, while Equation (3) produces new members away from the common viruses. The *intensity* in Equation (2) reduces the random perturbation that creates new viruses from the strong members. This will allow VOA to intensify exploitation in regions more likely to have a global optimum. In addition, the initial value for the *intensity* is set as one, which means that the random perturbation for strong and common viruses is the same in the early stages. The value of the *intensity* is increased by one whenever the performance (average objective function value) of the population does not improve between replications.

It is important to mention that VOA only allows feasible solutions, while infeasible solutions are discarded and replacements are regenerated.

3.3.3. *Updating and maintenance procedure*

Once the new viruses have been generated using the replication procedure, the corresponding objective function values are evaluated. The old and new viruses are then mixed together as a pooled population. As a result, only the strong viruses are kept ordered according to their objective function values.

3.3.3.1. *Updating the exploitation mechanism.* This step checks the population convergence, and VOA performance (*i.e.* average objective function value of the population) determines whether the exploitation has to be intensified. Exploitation is intensified by increasing the value of the variable *intensity* in Equation (2) by one if the average performance of VOA did not improve. This allows the algorithm to create new members closer to the stronger viruses, thus intensifying the exploitation in areas populated by strong viruses.

3.3.3.2. *Population maintenance mechanism.* The population maintenance mechanism, named antivirus, is activated by interaction between the viruses and the host cell, *i.e.* the antivirus is triggered at each replication, killing a given number of viruses according to Equation (1). Thus, the host cell is protected against viruses created during the replication process. Since the size of a single virus is on average one-thousandth that of an average cell (Campbell, Williamson, and Heyden 2006), if the total number of viruses inside the host cell exceeds 1000, the algorithm will reduce the size of the population to the amount set at parameter 1 (introduced in Section 3.2).

The antivirus mechanism evaluates the average objective function value as the performance of the virus population. Once verified, the number of members to be exterminated is determined by Equation (1); those common viruses performing worse than the aforementioned statistic (average objective function value) are individually eliminated from the population until the value given by Equation (1) is reached.

However, if the quantity of viruses exterminated is insufficient to reach the value given by Equation (1), members from the portion of viruses performing better than or equal to the average objective function are deleted at random until the target value is reached.

3.3.4. *Stopping criterion*

If population performance did not improve, exploitation is intensified and antivirus is implemented, followed by the confirmation of the stopping criterion. If the stopping criterion has not been reached, the replication counter is increased by one and the algorithm continues with the next replication stage; otherwise, the algorithm stops.

4. Computational results

4.1. *Benchmark functions*

In this study, a set of eight continuous benchmark functions from the literature (Suganthan *et al.* 2005; Yao, Liu, and Lin 1999) is used to verify VOA performance. The mathematical model and the properties of the benchmark functions are provided in Appendix A (see web link given at the end of this article). In addition, some of the benchmark functions are shifted and rotated to verify the robustness of the proposed VOA. This rotation is made by using an orthogonal matrix implementing Salomon's method (Salomon 1996) and the results are compared against those of several algorithms from the literature.

Table 2. Parameter setting recommended by design of experiments results for each benchmark function.

	Benchmark function tested							
Parameter	Sphere (f_1)	Rosenbrock (f_2)	Ackley (f_3)	Schwefel 2.22 (f_4)	Generalized Schwefel 2.26 (f_5)	Generalized Rastrigin (f_6)	Generalized Griewank (f_7)	Weierstrass (f_8)
No. of strong viruses	10	10	10	10	10	10	5	10
Growth rate (strong)	8	8	8	8	8	8	8	8
Growth rate (common)	2	2	2	2	2	2	2	2

4.2. *Parameter setting*

The VOA results are compared with corresponding results from the literature where the stopping criterion (number of function evaluations) is given, while the initial number of viruses is consistently set to 50 members. The methods selected for comparison are considered state-of-the-art optimization tools, and have been proven to be very powerful approaches when solving continuous domain functions.

For the setting of the three VOA parameters (*i.e.* number of strong viruses, growth rate of strong viruses and growth rate of common viruses), design of experiments (DoE) (Montgomery 2001) is adopted, where each parameter (*i.e. factor* in DoE) has three levels to be tested. The DoE results (summarized in Table 2) show that the same setting in the parameters is suggested for most of the functions. The growth rate of strong viruses is higher than that of common ones, indicating that the algorithm will focus more on exploiting regions where strong viruses are present. Meanwhile, the growth rate for common viruses is not zero, to avoid convergence at local optima.

4.3. *Performance comparison*

In this subsection, the performance and robustness of VOA are tested through a comprehensive and extensive comparison with variations of metaheuristic approaches, dimensionality and structure. The proposed VOA was implemented in C# language using an Intel Core i5 CPU 650 3.2 GHz with 2 GB of RAM.

4.3.1. *Comparison with artificial immune system variations*

Many similarities exist between the VOA and the widely known AIS, but critical differences are reviewed here. In VOA, the solution (value of each decision variable) represents the location of the virus in the multidimensional space, whereas in AIS they represent the type of antibody created by the immune system. The generation of new solutions (viruses or antibodies) differs between the two algorithms. AIS create clones, which are exact copies of the antibodies (solutions) currently contained in the population, that undergo a mutation-like mechanism to change their structure (*i.e.* the values of decision variables). In contrast, VOA implements Equations (2) and (3) depending on the type of virus (strong or common) to be replicated.

Another difference is the way in which the two methods maintain their populations. VOA implements an antivirus mechanism, which exterminates a number of members given by Equation (1) (see Section 3.1). AIS, on the other hand, needs to determine the affinity among antibodies, usually the Euclidean distance between solutions. Furthermore, AIS introduces a given percentage of randomly generated members in the population of antibodies whenever the maintenance mechanism is implemented, whereas this is not considered in the case of the VOA. Aside from the vaccine-enhanced AIS, all the algorithms were proposed by de Castro and Von Zuben (2002), de Castro and Timmis (2002), de França, Von Zuben, and de Castro (2005) and Woldemariam and Yen (2010). The source codes are available upon request from F. O. de França, one of the co-authors who proposed the dopt-Ainet. However, vaccine-enhanced AIS was coded by this study. All the approaches (including VOA) were implemented in C#; therefore, a change in the source code for the CLONALG was necessary.

The parameter settings of the competing AIS-based algorithms are summarized in Appendix B (see web link at the end of this article). The number of decision variables applied in this part of the study is 30, and the stopping criterion is set at 1×10^5 function evaluations. Table 3 provides detailed algorithm performance results averaged over 30 independent runs, and shows that VOA outperforms most of the competing algorithms with the smallest total sum of rank. Furthermore, VOA achieves the best performance on four out of the seven instances tested in this part of the study, where only the vaccine-enhanced AIS matches the performance of VOA with equal total sum of rank.

Moreover, when comparing the computational effort (CPU time in seconds), VOA is approximately four times faster than the fastest AIS-based algorithm (CLONALG) and nearly 15 times faster that the vaccine-enhanced AIS, which provides solution quality comparable to that of VOA. The relatively poor performance of the AIS variations in terms of CPU time could be attributed to the fact that the calculation of the affinity among antibodies accounts for most of the computational effort.

4.3.2. *Comparison in low-, medium- and high-dimensional instances*

To test the robustness of VOA over different dimension sizes, several continuous benchmark functions are used with small (30), medium (50 and 100) and large (500 and 1000) numbers of decision variables. Computational results are also compared with recent publications in which metaheuristic algorithms such as ABC, differential evolution, evolutionary programming, evolutionary strategies, GAs, HS, invasive weed optimization (IWO) and PSO were applied. The detailed computational results are presented as follows.

The first performance comparison was done according to Nasir *et al.* (2012), where the results are presented in terms of the average and standard deviation of the objective function over 25 independent runs. For a detailed explanation and parameter setting of each competing algorithm, please refer to Nasir *et al.* (2012). In addition, for the 30-dimension instances, newly developed algorithms are used to demonstrate VOA performance. The algorithms include a modified version of the ABC algorithm (Akay and Karaboga 2012), IWO, modified invasive weed optimization (M-IWO), differential invasive weed optimization (DIWO) (Basak, Maity, and Das 2013) and an enhanced invasive weed optimization algorithm (EIWO) (Ramezani, Ahangaran, and Yang 2013). These performance comparisons are summarized in Table 4. VOA significantly outperforms all of the competing algorithms. For the 30-dimension instances, VOA is not only a promising approach for low-dimensional instances, but also a novel and effective approach which does not entail hybridization with other methods and holds significant potential for other applications.

For the medium-dimensional instances, two recent articles published by Das *et al.* (2011) and Gong *et al.* (2011) were used as a basis for comparison. The benchmark functions were tested with 50 dimensions for Das *et al.* (2011) and 100 dimensions for Gong *et al.* (2011). For the performance measure the mean and standard deviation of the best-of-run error after 50 independent runs is applied. The best-of-run error corresponds to the absolute difference between

Table 3. Performance comparison between the virus optimization algorithm (VOA) and artificial immune system (AIS)-based algorithms.

		Benchmark function used (dimensions = 30)							
Algorithm	Measure	Sphere (f_1)	Generalized Rosenbrock (f_2)	Ackley (f_3)	Schwefel 2.22 (f_4)	Generalized Schwefel 2.26 (f_5)	Generalized Rastrigin (f_6)	Generalized Griewank (f_7)	Sum of rank
VOA	Avg. obj.	**2.0367E − 54**	1.4522E + 01	1.2337E − 12	**2.9042E − 30**	**− 1.0554E + 04**	**0.0000E + 00**	2.9088E − 01	12
	StDev.	**2.0962E − 54**	2.5188E + 00	2.1073E − 20	**2.2143E − 30**	**4.3363E + 02**	**0.0000E + 00**	1.7354E − 01	
	Rank	**1st**	3rd	2nd	**1st**	**1st**	**1st**	3rd	
	Time (s)	**2.4570**	3.9203	2.9546	**3.0126**	**2.2365**	**1.9987**	2.1253	
dopt-aiNet	Avg. obj.	2.3654E − 20	1.2777E + 01	6.3266E − 11	1.5980E − 21	− 9.3693E + 03	0.0000E + 00	1.5990E − 02	15
	StDev.	6.3265E − 22	5.5924E + 00	3.6200E − 12	5.3690E − 20	3.6233E + 02	0.0000E + 00	2.3600E − 01	
	Rank	3rd	2nd	3rd	3rd	2nd	1st	1st	
	Time (s)	20.9831	23.2089	24.8766	33.1771	40.2587	33.1156	24.0603	
Vaccine-enhanced AIS	Avg. obj.	9.3265E − 44	**3.6778E + 00**	**5.2365E − 13**	6.3265E − 25	− 8.6954E + 03	**0.0000E + 00**	3.2650E − 02	12
	StDev.	6.3265E − 42	**1.8365E + 00**	**3.6600E − 13**	3.2655E − 24	1.2363E + 02	**0.0000E + 00**	2.2254E − 02	
	Rank	2nd	**1st**	**1st**	2nd	3rd	**1st**	2nd	
	Time (s)	30.3508	**37.5806**	**35.2584**	61.1538	61.5876	48.9599	40.8133	
CLONALG	Avg. obj.	1.0333E + 01	2.0302E + 02	1.2426E + 01	2.3665E − 05	− 7.6425E + 03	4.2554E + 02	2.1562E + 01	33
	StDev.	2.3569E − 01	2.0250E + 01	1.2232E + 00	1.5006E − 06	2.3633E + 02	2.3626E + 00	3.2578E + 00	
	Rank	5th	5th	5th	5th	5th	3rd	5th	
	Time (s)	8.3452	9.1815	8.9324	8.0590	6.1286	9.0577	9.4777	
Opt-aiNet	Avg. obj.	5.9870E − 03	2.5065E + 01	1.2046E − 02	1.2146E − 06	− 8.0063E + 03	9.9637E − 01	7.2700E + 00	26
	StDev.	5.7840E − 03	1.5310E + 00	9.3626E − 02	8.9659E − 06	1.2226E + 02	1.2399E − 02	2.3570E − 01	
	Rank	4th	4th	4th	4th	4th	2nd	4th	
	Time (s)	19.0064	23.0607	23.0902	31.9511	39.1859	31.9631	22.6047	

Table 4. Results comparison of low-dimensional instances (dimensions = 30).

		Benchmark function used (dimensions = 30)				
Algorithm	Measure	Sphere (f_1)	Generalized Rosenbrock (f_2)	Ackley (f_3)	Schwefel 2.22 (f_4)	Sum of rank
VOA	Avg. obj.	**7.7280E − 172**	8.8068E + 00	**2.0985E − 16**	**8.1487E − 93**	**6**
	StDev.	**2.7799E − 150**	3.4697E + 00	**2.1073E − 18**	**1.2473E − 92**	
	Rank	**1st**	3rd	**1st**	**1st**	
PSOlocal	Avg. obj.	1.2040E − 07	2.5950E + 01	8.1660E − 05	2.1550E − 06	40
	StDev.	1.3180E − 07	5.6940E + 00	2.5120E − 05	1.0066E − 06	
	Rank	11th	8th	10th	11th	
UPSO	Avg. obj.	4.5680E − 51	2.1180E + 01	4.4400E − 15	4.7180E − 30	15
	StDev.	3.0070E − 51	2.4580E + 00	0.0000E + 00	2.8600E − 30	
	Rank	3rd	6th	3rd	3rd	
FIPS	Avg. obj.	3.9660E − 06	2.6870E + 01	4.4290E − 04	1.9960E − 04	44
	StDev.	1.4570E − 06	1.5940E + 01	1.2710E − 04	3.3900E − 05	
	Rank	12th	9th	11th	12th	
DMSPSO	Avg. obj.	4.3770E − 45	3.9660E + 01	8.9680E − 10	9.4780E − 28	29
	StDev.	1.3420E − 44	2.6130E + 01	1.0490E + 01	2.6640E − 27	
	Rank	6th	12th	7th	4th	
CLPSO	Avg. obj.	7.7640E − 11	1.4340E + 01	3.7270E − 06	1.9050E − 07	33
	StDev.	2.1840E − 11	5.7090E + 00	6.6680E − 07	2.9400E − 08	
	Rank	10th	4th	9th	10th	
DNLPSO	Avg. obj.	9.4450E − 74	4.7610E + 00	1.9030E + 01	7.0160E − 09	25
	StDev.	9.1320E − 32	2.8610E + 00	0.0000E + 00	7.0160E − 09	
	Rank	2nd	2nd	12th	9th	
Modified ABC	Avg. obj.	6.4910E − 18	2.7100E + 01	1.1336E − 15	1.2351E − 16	28
	StDev.	7.1052E − 19	2.3263E + 01	1.3019E − 14	2.6609E − 17	
	Rank	9th	10th	2nd	7th	
IWO	Avg. obj.	1.3332E − 36	2.3711E + 01	1.2725E − 07	3.2721E − 15	31
	StDev.	8.6519E − 37	4.8982E + 00	1.2096E − 07	2.8746E − 15	
	Rank	8th	7th	8th	8th	
M-IWO	Avg. obj.	2.5673E − 40	2.9481E + 01	2.3315E − 12	6.2739E − 19	30
	StDev.	2.4531E − 40	2.9232E + 01	8.8048E − 13	4.8962E − 19	
	Rank	7th	11th	6th	6th	
EIWO	Avg. obj.	3.5225E − 45	1.5223E + 01	1.0691E − 12	9.2466E − 22	20
	StDev.	2.9045E − 45	8.6333E + 00	8.0316E − 13	8.2354E − 22	
	Rank	5th	5th	5th	5th	
DIWO	Avg. obj.	5.8964E − 50	**2.1665E − 02**	5.5713E − 13	1.9622E − 40	11
	StDev.	8.2909E − 51	**8.5004E − 03**	3.7185E − 13	2.8911E − 41	
	Rank	4th	**1st**	4th	2nd	

the best value found during a particular run and the true optimum of the function under study. The computers used in the articles presented above were not specified. The detailed results for the performance of the algorithms implemented in this part of the study are summarized in Tables 5 and Table 6.

The results in Tables 5 and 6 indicate that VOA outperforms all the algorithms from Das *et al.* (2011) and Gong *et al.* (2011) with the smallest rank sum. Note that VOA performs the best in five out of six and five out of the seven instances, respectively, for the 50- and 100-dimensional benchmark functions. This outstanding performance suggests that VOA is a viable approach for solving this type of continuous domain benchmark function. Furthermore, it is important to note that VOA is competitive with algorithmic tools presented in the literature on medium-dimensional instances.

For the high-dimensional instances, VOA is compared with the methods in Yang, Tang, and Yao (2008). Performance is measured in terms of the mean objective function value over 25 independent runs, and Table 7 and Table 8 compare results with the competing algorithms. When

Table 5. Results comparison of medium-dimensional instances (dimensions = 50).

		Benchmark function used (dimensions = 50)						
Algorithm	Measure	Sphere (f_1)	Generalized Rosenbrock (f_2)	Ackley (f_3)	Generalized Rastrigin (f_6)	Generalized Griewank (f_7)	Weierstrass (f_8)	Sum of rank
VOA	Avg. error	0.0000E + 00	4.8590E + 01	4.4409E – 16	0.0000E + 00	0.0000E + 00	0.0000E + 00	11
	StDev.	0.0000E + 00	4.8712E – 01	0.0000E + 00	0.0000E + 00	0.0000E + 00	0.0000E + 00	
	Rank	1st	6th	1st	1st	1st	1st	
CPSO-H6	Avg. error	5.7534E – 11	3.7263E – 01	1.7725E – 09	7.5712E – 01	5.6548E – 02	8.5829E – 14	24
	StDev.	2.1094E – 11	1.8376E – 01	2.4893E – 10	1.2439E – 01	2.3031E – 02	1.6792E – 09	
	Rank	5th	2nd	5th	5th	5th	2nd	
LEA	Avg. error	6.8590E – 15	4.9274E – 01	5.0520E – 15	2.5358E – 17	6.5132E – 15	2.9890E – 12	17
	StDev.	4.9847E – 15	8.7231E – 01	7.6380E – 14	6.9563E – 16	1.7965E – 13	7.0836E – 10	
	Rank	2nd	3rd	4th	2nd	3rd	3rd	
ALEP	Avg. error	6.3244E – 04	2.8263E + 01	3.7160E – 02	6.0499E + 00	1.7382E – 01	9.6648E – 05	38
	StDev.	5.3432E – 04	4.9826E + 00	9.3228E – 02	8.3452E – 01	4.0930E – 02	2.3310E – 04	
	Rank	7th	5th	7th	7th	6th	6th	
DE/rand/1/bin	Avg. error	9.5340E – 15	1.9325E + 00	1.1253E – 15	5.7362E – 12	3.4837E – 14	1.8092E + 01	25
	StDev.	3.2981E – 15	2.8271E + 00	3.4536E – 14	3.8273E – 14	2.6367E – 15	6.1253E + 00	
	Rank	3rd	4th	2nd	3rd	4th	9th	
IHS	Avg. error	8.3392E – 01	6.2761E + 04	1.1802E + 01	2.3468E + 02	1.9238E + 02	1.6353E – 03	52
	StDev.	4.8337E – 02	3.8157E + 03	9.4657E – 01	2.8233E + 01	3.1569E + 01	8.2304E – 02	
	Rank	9th	9th	9th	9th	9th	7th	
GHS	Avg. error	8.9437E – 02	4.8610E + 04	8.7631E + 00	7.5463E + 01	4.9728E + 01	9.5278E – 01	48
	StDev.	1.0433E – 02	2.7963E + 03	7.9245E – 01	4.9806E + 00	9.8384E + 00	4.4272E – 01	
	Rank	8th	8th	8th	8th	8th	8th	
MHS	Avg. error	3.3574E – 04	6.3198E + 02	9.3747E – 06	9.6429E – 01	1.0609E + 00	7.6687E – 05	37
	StDev.	2.4474E – 04	4.8722E + 01	3.1042E – 06	5.5854E – 01	3.4546E – 07	6.7367E – 05	
	Rank	6th	7th	6th	6th	7th	5th	
EHS	Avg. error	3.0610E – 12	1.2094E – 03	2.0345E – 15	6.3927E – 12	5.8924E – 16	6.0189E – 09	18
	StDev.	4.2572E – 12	3.8237E – 04	6.7564E – 16	4.0475E – 12	6.0042E – 15	3.8271E – 07	
	Rank	4th	1st	3rd	4th	2nd	4th	

Table 6. Results comparison of medium-dimensional instances (dimensions = 100).

		Benchmark function used (dimensions = 100)							
Algorithm	Measure	Sphere (f_1)	Generalized Rosenbrock (f_2)	Ackley (f_3)	Schwefel 2.22 (f_4)	Generalized Schwefel 2.26 (f_5)	Generalized Rastrigin (f_6)	Generalized Griewank (f_7)	Sum of rank
VOA	Avg. error	0.0000E + 00	7.8574E + 01	4.3341E – 16	0.0000E + 00	2.3710E + 04	0.0000E + 00	0.0000E + 00	15
	StDev.	0.0000E + 00	3.0058E – 01	0.0000E + 00	0.0000E + 00	0.0000E + 00	0.0000E + 00	0.0000E + 00	
	Rank	1st	4th	1st	1st	6th	1st	1st	
jDE	Avg. error	2.0900E – 20	9.2000E + 01	1.7300E – 11	1.8200E – 12	2.8100E – 08	6.0100E + 00	0.0000E + 00	27
	StDev.	9.2700E – 21	1.4100E + 01	3.1500E – 12	4.3000E – 13	2.3000E – 08	2.3600E + 00	0.0000E + 00	
	Rank	6th	6th	5th	6th	1st	2nd	1st	
JADE-wo	Avg. error	5.1300E – 62	4.9600E + 01	7.6900E – 15	5.1900E – 36	3.9400E + 03	1.0300E + 02	8.8700E – 04	22
	StDev.	4.8200E – 62	1.1500E + 01	0.0000E + 00	7.1200E – 36	2.7500E + 02	3.9500E + 00	3.2500E – 03	
	Rank	4th	3rd	2nd	4th	3rd	3rd	3rd	
JADE-w	Avg. error	1.2100E – 85	2.7700E + 01	7.8400E – 15	9.2000E – 42	9.1100E + 03	1.8200E + 02	0.0000E + 00	22
	StDev.	2.2700E – 85	6.8100E + 00	7.0300E – 16	2.9600E – 41	4.1800E + 02	8.4400E + 00	0.0000E + 00	
	Rank	3rd	2nd	3rd	3rd	4th	6th	1st	
SaDE	Avg. error	1.0900E – 27	8.4900E + 01	1.0500E – 14	1.0900E – 15	1.8900E + 01	1.0500E + 02	2.9600E – 04	27
	StDev.	6.6500E – 28	1.0400E + 01	1.7600E – 15	2.1000E – 16	3.5100E + 01	4.8400E + 00	1.4600E – 03	
	Rank	5th	5th	4th	5th	2nd	4th	2nd	
SaJADE	Avg. error	1.6200E – 92	2.4500E + 01	7.6900E – 15	4.0300E – 51	9.0400E + 03	1.6700E + 02	0.0000E + 00	18
	StDev.	2.9200E – 92	1.4300E + 00	0.0000E + 00	1.4100E – 50	3.8800E + 02	7.5900E + 00	0.0000E + 00	
	Rank	2nd	1st	2nd	2nd	5th	5th	1st	

Table 7. Results comparison of high-dimensional instances (dimensions = 500).

		Benchmark function used (dimensions = 500)							
Algorithm	Measure	Sphere (f_1)	Generalized Rosenbrock (f_2)	Ackley (f_3)	Schwefel 2.22 (f_4)	Generalized Schwefel 2.26 (f_5)	Generalized Rastrigin (f_6)	Generalized Griewank (f_7)	Sum of rank
VOA	Avg. obj.	0.0000E + 00	4.9832E + 02	4.4409E – 16	0.0000E + 00	– 1.9325E + 05	0.0000E + 00	0.0000E + 00	12
	StDev.	0.0000E + 00	3.1111E – 01	0.0000E + 00	0.0000E + 00	1.5194E + 03	0.0000E + 00	0.0000E + 00	
	Rank	1st	2nd	1st	1st	5th	1st	1st	
SaNSDE	Avg. obj.	2.4100E – 11	1.3300E + 03	7.8800E + 00	5.2700E – 02	– 2.0180E + 05	2.8400E + 02	1.8200E – 01	31
	Rank	4th	4th	5th	5th	4th	4th	5th	
FEPCC	Avg. obj.	4.9000E – 08	–[a]	5.7000E – 04	1.3000E – 03	– 2.0932E + 05	1.4300E – 01	2.9000E – 02	27
	Rank	5th	5th	4th	4th	3rd	2nd	4th	
DECC-O	Avg. obj.	2.2800E – 21	6.6400E + 02	1.8600E – 11	3.7700E – 10	– 2.0949E + 05	1.7600E + 01	5.0200E – 16	20
	Rank	3rd	3rd	3rd	3rd	2nd	3rd	3rd	
DECC-G	Avg. obj.	6.3300E – 27	4.9200E + 02	9.1300E – 14	5.9500E – 15	– 2.0949E + 05	0.0000E + 00	4.4000E – 16	11
	Rank	2nd	1st	2nd	2nd	1st	1st	2nd	

Note: [a]The result was not provided in the study.

Table 8. Results comparison of high-dimensional instances (dimensions = 1000).

		Benchmark function used (dimensions = 1000)							
Algorithm	Measure	Sphere (f_1)	Generalized Rosenbrock (f_2)	Ackley (f_3)	Schwefel 2.22 (f_4)	Generalized Schwefel 2.26 (f_5)	Generalized Rastrigin (f_6)	Generalized Griewank (f_7)	Sum of rank
VOA	Avg. obj.	0.0000E + 00	9.9839E + 02	4.4409E – 16	0.0000E + 00	– 3.4407E + 05	0.0000E + 00	0.0000E + 00	12
	StDev.	0.0000E + 00	1.1547E – 01	0.0000E + 00	0.0000E + 00	8.5270E + 03	0.0000E + 00	0.0000E + 00	
	Rank	1st	2nd	1st	1st	5th	1st	1st	
SaNSDE	Avg. obj.	6.9700E + 00	3.3100E + 03	1.1200E + 01	1.2400E + 00	– 3.7299E + 05	8.6900E + 02	4.8000E – 01	32
	Rank	5th	4th	5th	4th	4th	5th	5th	
FEPCC	Avg. obj.	5.4000E – 08	–[a]	9.5000E – 04	2.6000E – 03	– 4.1862E + 05	3.1300E – 01	2.5000E – 02	26
	Rank	4th	5th	4th	3rd	3rd	3rd	4th	
DECC-O	Avg. obj.	1.7700E – 20	1.4800E + 03	4.3900E – 11	INF	– 4.1898E + 05	3.1200E + 01	2.0400E – 15	23
	Rank	3rd	3rd	3rd	5th	2nd	4th	3rd	
DECC-G	Avg. obj.	2.1700E – 25	9.8700E + 02	2.2200E – 13	5.3700E – 14	– 4.1898E + 05	3.5500E – 16	1.0100E – 15	12
	Rank	2nd	1st	2nd	2nd	1st	2nd	2nd	

Note: [a]The result was not provided in the study.

Table 9. Results comparison with algorithms considering both shifted and rotated factors.

		Shifted and rotated benchmark function used (dimensions = 30)						
Algorithm	Measure	Rosenbrock	Ackley	Griewank	Weierstrass	Rastrigin	Generalized Schwefel 2.26	Sum of rank
VOA	Avg. error	2.2828E + 01	0.0000E + 00	0.0000E + 00	0.0000E + 00	0.0000E + 00	3.5994E + 03	22
	StDev.	4.2697E – 02	0.0000E + 00	0.0000E + 00	0.0000E + 00	0.0000E + 00	6.8343E + 02	
	Rank	7th	1st	1st	1st	1st	11th	
GA	Avg. error	2.9000E + 01	2.6200E + 00	7.1000E – 01	1.6200E + 01	9.3000E + 01	2.7000E + 03	61
	StDev.	6.4000E – 01	2.4600E + 00	1.3000E – 01	2.0200E + 00	3.7900E + 01	7.5400E + 02	
	Rank	12th	9th	11th	11th	11th	7th	
MA	Avg. error	1.1700E + 02	3.6400E + 00	3.4200E – 02	2.1000E + 01	1.0300E + 02	–[a]	70
	StDev.	1.4600E + 01	3.6400E + 00	3.4200E – 02	2.1000E + 01	2.0700E + 01	–[a]	
	Rank	13th	10th	10th	12th	12th	13th	
PSO-w	Avg. error	2.7900E + 01	1.4200E + 00	1.5700E – 02	7.8700E + 00	5.4300E + 01	2.3300E + 03	52
	StDev.	1.3900E + 01	7.3300E – 01	1.8000E – 02	2.8500E + 00	1.4800E + 01	6.2100E + 02	
	Rank	11th	8th	9th	10th	10th	4th	
CLPSO	Avg. error	2.3100E + 01	8.6600E – 06	8.3100E – 07	3.3100E + 00	3.4500E + 01	2.0500E + 03	37
	StDev.	2.0900E + 00	2.4000E – 05	2.3400E – 06	1.3900E + 00	5.8100E + 00	3.1400E + 02	
	Rank	8th	6th	8th	6th	6th	3rd	
Rand-BFGS	Avg. error	8.0600E – 11	1.9100E + 01	4.7600E – 13	3.4700E + 01	2.8600E + 02	5.6200E + 03	56
	StDev.	1.4700E – 10	1.9200E – 01	4.9100E – 13	1.9800E + 00	3.9200E + 01	4.2100E + 02	
	Rank	4th	11th	3rd	13th	13th	12th	
DMS-L-PSO	Avg. error	1.8100E – 10	3.4000E – 14	5.6000E – 14	3.3700E – 01	3.1900E + 01	3.2800E + 03	27
	StDev.	9.9100E – 11	1.5000E – 14	3.9500E – 13	6.9300E – 01	5.4700E + 00	5.9200E + 02	
	Rank	5th	3rd	2nd	3rd	5th	9th	
PSO-w-BFGS	Avg. error	3.5300E – 14	6.1600E – 08	5.4600E – 13	4.9600E + 00	5.0000E + 01	3.5900E + 03	38
	StDev.	3.7100E – 14	4.3600E – 07	4.6800E – 13	2.4000E + 00	1.2000E + 01	7.1300E + 02	
	Rank	2nd	5th	4th	8th	9th	10th	
CLPSO-BFGS	Avg. error	5.5100E – 14	2.3100E – 14	6.1400E – 13	1.0300E + 00	1.3000E + 01	2.4200E + 03	21
	StDev.	5.9700E – 14	7.1700E – 15	5.6500E – 13	1.4600E + 00	1.2400E + 01	4.0000E + 02	
	Rank	3rd	2nd	5th	4th	2nd	5th	
PSO-w-NBFGS	Avg. error	7.9700E – 02	1.5500E – 01	8.2700E – 08	5.4100E + 00	4.4900E + 01	2.9300E + 03	43
	StDev.	5.6400E – 01	4.3100E – 01	4.5900E – 08	2.3100E + 00	1.1000E + 01	7.6000E + 02	
	Rank	6th	7th	6th	9th	7th	8th	
CLPSO-NBFGS	Avg. error	1.4100E – 14	1.9700E – 12	1.9600E – 07	2.2000E + 00	2.7800E + 01	2.0200E + 03	22
	StDev.	1.3400E – 14	1.0100E – 13	1.3300E – 07	1.9000E + 00	5.6300E + 00	3.5500E + 02	
	Rank	1st	4th	7th	5th	3rd	2nd	
DMS-L-ASA	Avg. error	2.5000E + 01	0.0000E + 00	0.0000E + 00	6.1000E – 01	4.6300E + 01	2.6500E + 03	27
	StDev.	1.8000E – 01	0.0000E + 00	0.0000E + 00	1.0900E + 00	5.3200E + 00	6.7400E + 02	
	Rank	9th	1st	1st	2nd	8th	6th	
CLPSO-ASA	Avg. error	2.5400E + 01	0.0000E + 00	0.0000E + 00	4.4300E + 00	3.1600E + 01	1.9300E + 03	24
	StDev.	1.0000E + 00	0.0000E + 00	0.0000E + 00	2.2100E + 00	7.3300E + 00	3.5800E + 02	
	Rank	10th	1st	1st	7th	4th	1st	

Note: [a]The result was not provided in the study.

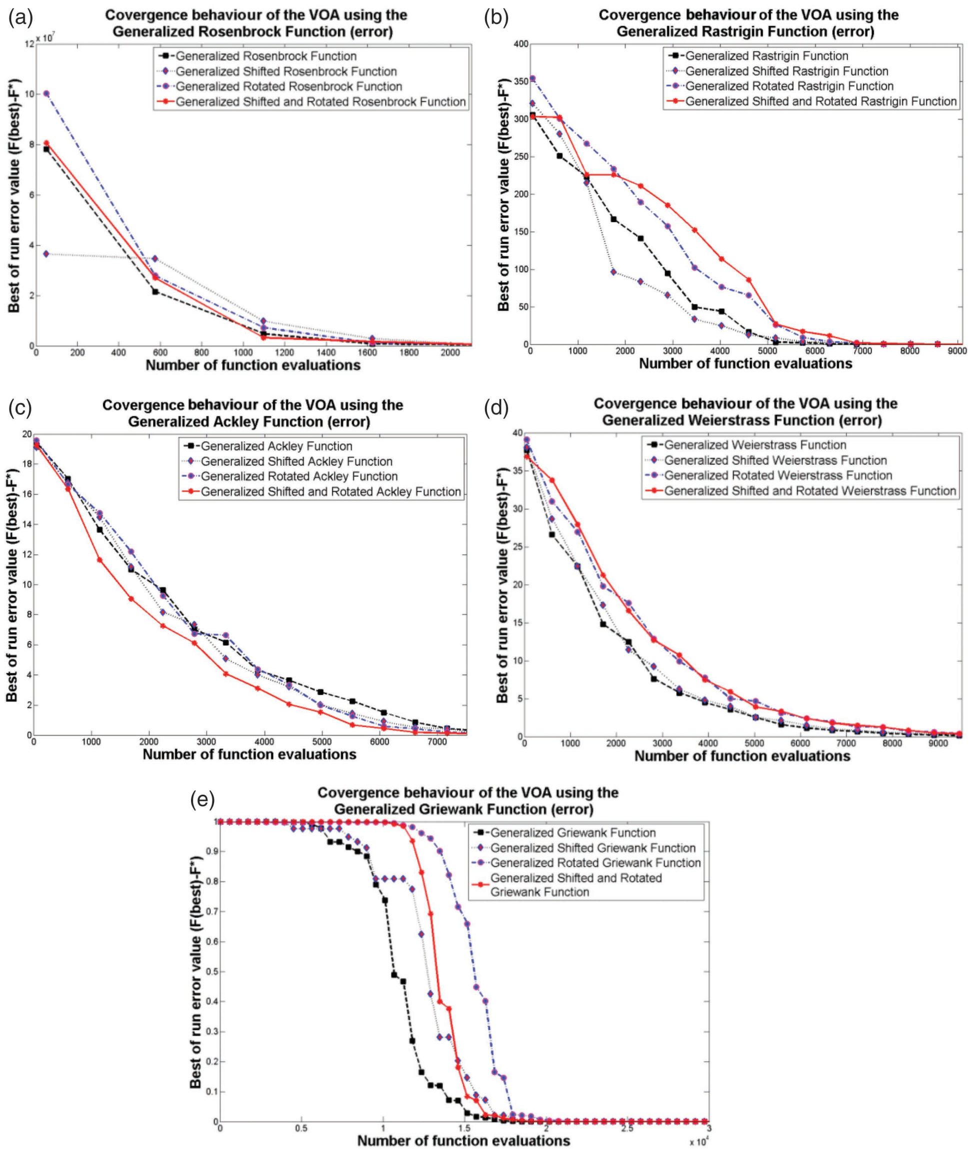


Figure 3. Convergence behaviour of the virus optimization algorithm (VOA) using (a) the Rosenbrock function, (b) the Rastrigin function, (c) the Ackley function, (d) the Weierstrass function, and (e) the Griewank function.

solving large-sized instances (*i.e.* 500 and 1000 variables each), VOA still provides stable performance and outperforms most of the competing algorithms from the literature. In this group, only DECC-G is competitive with VOA. VOA performs the best in five out of seven instances in both the 500- and 1000-dimensional instances.

4.3.3. *Test on benchmarks with rotated and shifted factors*

The complexity of the benchmark functions increased when the structure was adjusted subject to shifted and rotated factors. Thus, the aforementioned is hoped to provide insight knowledge

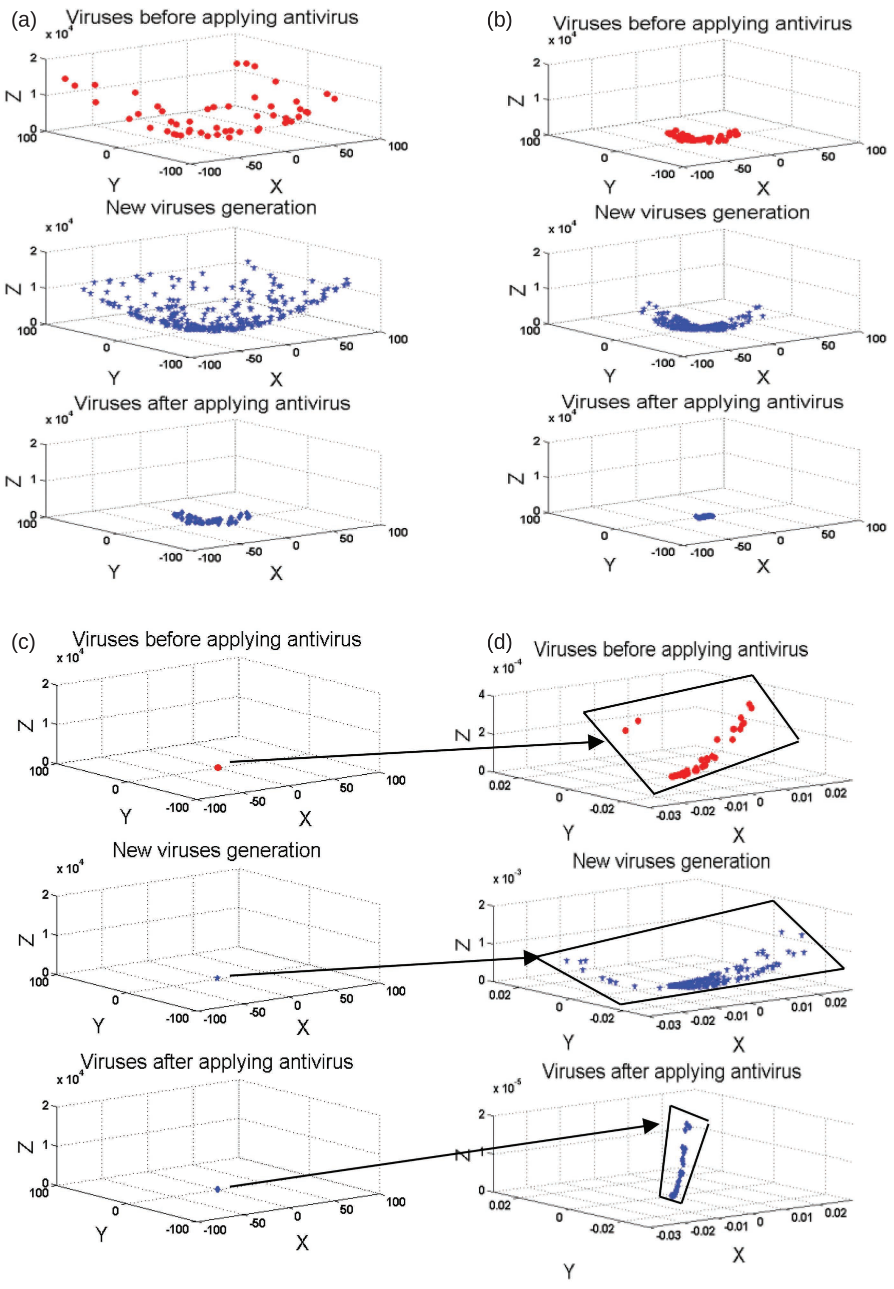


Figure 4. Convergence behaviour of the virus optimization algorithm (VOA) at (a) iteration 1, (b) iteration 10, (c) iteration 20, and (d) iteration 20 with modification of the scale to zoom in.

on the optimization behaviour of VOA. For this part of the study, VOA is compared with algorithmic tools introduced by Li *et al.* (2011), where the stopping criterion was 1.8×10^5 function evaluations for 30-dimension instances. The performance was measured in terms of the mean and variance of the best-of-run error among 50 independent runs. The comparison results are shown in Table 9. For the shifting of the functions, a constant value (*f*_bias) is added to the objective function, leading to the form presented in Equation (4). The *f*_bias values for each function used in Li *et al.* (2011) are specified in Appendix C (see web link at the end of this article). The rotation is performed by multiplying the original vector containing variables *x* with an orthogonal matrix **M**, resulting in a new vector containing the variables *y*, as in Equation (5).

$$f'(\mathbf{x}) = f(\mathbf{x}) + f\_\text{bias} \tag{4}$$

$$\mathbf{y} = \mathbf{M}\mathbf{x} \tag{5}$$

In Equation (5), the matrix **M** is obtained using Salomon's method (Salomon 1996) and the vector **y** is the variables after rotation; that is, the variable y_i substitutes the original variables x_i, as in Equation (6). Except for the Schwefel Problem 2.26, all the benchmark functions have the same mathematical formulation specified in Appendix A. The global optimum of each benchmark function remains the same when only rotation is applied; however, when a shifted term is implemented the global optimum is shifted by the *f_bias* value, as follows:

$$f'(\mathbf{y}) = f(\mathbf{y}) + f\_\text{bias} \tag{6}$$

Table 9 shows that VOA still provides stable performance when the benchmark functions are subject to a shifted and rotated factor. In this group, when rotated and shifted terms are both implemented, only two methods (CLPSO-BFGS and CLPSO-NBFGS) remain competitive with VOA. Once again, VOA not only outperforms most of the methods in the literature, but also shows outstanding results when the benchmark functions are shifted and/or rotated. Through different sizes and types of instances, the performance of VOA suggests that it is a highly effective optimization tool for solving continuous problems.

Lastly, Figure 3 shows the convergence behaviour of the VOA solving different benchmark functions with 30 dimensions, where its superior performance can be appreciated. In addition, it can be observed that VOA is capable of converging to the global or near-global optimum, and the rotated and shifted factors of the benchmark functions do not represent a major obstacle for VOA. This suggests that the proposed method is robust when considering shifted and rotated factors.

Figure 4 illustrates how the VOA behaves during the replication and population maintenance procedures using the sphere function. At replication 1, all the viruses are spread out in the solution space (*i.e.* inside the host cell). However, after 10 replications, the virus population starts to gather around the region where the global optimum is located. As shown in Figure 4(c), after the 20th replication, what seems to be a single point is in fact a group of viruses located at or close to the global optimum. Figure 4(d) provides a better view of the scattering behaviour of these viruses.

5. Conclusions

This study introduces a newly developed metaheuristic named the virus optimization algorithm (VOA), inspired by the behaviour of viruses attacking a living cell (host). This idea is shown to be very effective at solving continuous domain problems. A similar algorithmic tool, also inspired from biology, named artificial immune systems (AIS), was compared against VOA, but these two approaches are only similar in terms of terminology, since they do not perform or behave in the same way during the optimization process.

The proposed VOA's sensitivity to parameter setting was observed using DoE, with VOA using the same setting for most benchmark functions. The results over test instances with different dimensions show that, despite its novelty, the VOA is competitive with widely known and well-developed metaheuristics such as ABC, AIS, differential evolution, evolutionary programming, evolutionary strategy, GA, HS, PSO and IWO. Test results showed that the proposed VOA performs well among different sizes of instances, including those of high dimensionality (500 or 1000 variables), which approximate real-problem applications.

For the benchmark functions subject to shifted and rotated factors, VOA also showed competitive performance in terms of average objective function value, achieving the best results in most of the tested instances. The convergence behaviour of the VOA was insensitive to the consideration of shifted and/or rotated factors, suggesting that the proposed VOA is sufficiently robust when solving continuous optimization problems with the different dimensions and factors considered in this study.

The proposed algorithm is very easy to parallelize, and can thus accelerate implementation on computer clusters, where each node is responsible for exploration and/or exploitation depending on the type of virus(es) present. Therefore, computational load can be further reduced when dealing with a dynamic population such as the one in VOA.

Future studies should determine whether the proposed algorithm can be used for solving discrete problems as well as reducing the number of controllable parameters. In addition, the rule of classifying the viruses (objective function value) into strong or common seems overly simplistic. Although the VOA showed outstanding results when compared with recent state-of-the-art metaheuristic methods, a better classification rule is something that could be considered in the future.

Supplemental data

Supplemental data for this article can be accessed at http://dx.doi.org/10.1080/0305215X.2014.994868.

Funding

This work was partially supported by the Ministry of Science and Technology, Taiwan [grant NSC100-2628-E-155-004-MY3] and by the Innovation Center for Big Data and Digital Convergence at Yuan Ze University.

References

Akay, B., and D. Karaboga. 2012. “A Modified Artificial Bee Colony Algorithm for Real-Parameter Optimization.” *Information Sciences* 192: 120–142.

Balicki, J. 2009. “An Adaptive Quantum-based Multiobjective Evolutionary Algorithm for Efficient Task Assignment in Distributed Systems.” The 13th WSEAS International Conference on Computers, 417–422.

Basak, A., D. Maity, and S. Das. 2013. “A Differential Invasive Weed Optimization Algorithm for Improved Global Numerical Optimization.” *Applied Mathematics and Computation* 219 (12): 6645–6668.

Campbell, N. A., and J. B. Reece. 2001. *Biology*. 6th ed. San Francisco: Benjamin Cummings.

Campbell, N. A., B. Williamson, and R. J. Heyden. 2006. *Biology: Exploring Life*. Boston: Pearson Prentice Hall.

de Castro, L. N., and J. Timmis, 2002. “An Artificial Immune Network for Multimodal Function Optimization.” *IEEE Congress on Evolutionary Computation (CEC’02)* 1: 669–674

de Castro, L. N., and F. J. Von Zuben. 2002. “Learning and Optimization Using the Clonal Selection Principle.” *IEEE Transactions on Evolutionary Computation* 6: 239–251.

Cuevas, J. R., H. J. Wang, Y. C. Lai, and Y. C. Liang. 2009. “Virus Optimization Algorithm: A Novel Meta-heuristic for Solving Continuous Optimization Problems.” *The 10th Asia Pacific Industrial Engineering & Management System Conference*, 2166–2174.

Das, S., A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi. 2011. “Exploratory Power of the Harmony Search Algorithm: Analysis and Improvements for Global Numerical Optimization.” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 41: 89–106.

Davidson, W. 2010. *The Molecular Expressions*. http://micro.magnet.fsu.edu/

Dorigo, M. 1992. “Optimization, Learning and Natural Algorithms.” Ph.D. thesis, Politecnico di Milano, Milano, Italy.

Farmer, J. D., N. H. Packard, and A. Perelson. 1986. “The Immune System, Adaptation and Machine Learning.” *Physica D* 22: 187–204.

Flint, S. J., L. W. Enquist, R. M. Krug, V. R. Racaniello, and A. M. Skalka. 1999. *Principles of Virology: Molecular Biology, Pathogenesis, and Control*. 1st ed. Washington, DC: ASM Press.

de França, F. O., F. J. Von Zuben, and L. N. de Castro. 2005. “An Artificial Immune Network for Multimodal Function Optimization on Dynamic Environments.” *Genetic Evolutionary Computation Conference*, 289–296.

Gong, W., Z. Cai, C. Ling, and H. Li. 2011. "Enhanced Differential Evolution with Adaptive Strategies for Numerical Optimization." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 41: 397–413.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. 19th ed. Ann Arbor, MI: University of Michigan Press.

Kennedy, J., and R. Eberhart. 1995. "Particle Swarm Optimization." *IEEE International Conference on Neural Networks* 1942–1948.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi. 1983. "Optimization by Simulated Annealing." *Science* 220: 671–680.

Lee, K. S., and Z. W. Geem. 2005. "A New Meta-Heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice." *Computer Methods in Applied Mechanics and Engineering* 194: 3902–3933.

Li, P., K. Song, and E. Yang. 2010. "Model and Algorithm of Neural Networks with Quantum Gated Nodes." *Neural Network World* 20: 189–206.

Li, S., M. Tan, I. W. Tsang, and J. T-Y. Kwok. 2011. "A Hybrid PSO-BFGS Strategy for Global Optimization of Multimodal Functions." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 41: 1003–1014.

Liu, Y., and K. M. Passino. 2002. "Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors." *Journal of Optimization Theory and Applications* 115: 603–628.

Madigan, T., J. Martinko, and J. Parker. 2006. *Brock Biology of Microorganisms*. 11th ed. San Francisco: Pearson Prentice Hall.

Montgomery, D. 2001. *Design and Analysis of Experiments*. 5th ed. New York: John Wiley & Sons.

Nasir, M. d., S. Das, D. Maity, S. Sengupta, U. Halder, and P. N. Suganthan. 2012. "A Dynamic Neighborhood Learning Based Particle Swarm Optimizer for Global Optimization." *Information Sciences* 209: 16–36.

Pham, D. T., A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. 2005. "The Bees Algorithm." Technical Note, Manufacturing Engineering Centre, Cardiff University, UK.

Ramezani, P., M. Ahangaran, and X. S. Yang. 2013. "Constrained Optimisation and Robust Function Optimisation with EIWO." *International Journal of Bio-Inspired Computation* 5 (2): 84–98.

Riss, T. L., and R. A. Moravec. 2004. "Use of Multiple Assay Endpoints to Investigate the Effects of Incubation Time, Dose of Toxin, and Plating Density in Cell-Based Cytotoxicity Assays." *Assay Drug Development Technology* 2: 51–62.

Roulston, A., R. C. Marcellus, and P. E. Branton. 1999. "Viruses and Apoptosis." *Annual Review of Microbiology* 53: 577–628.

Russell, P. J. 2001. *iGenetics: A Mendelian Approach*. 1st ed. San Francisco: Benjamin Cummings.

Salomon, R. 1996. "Reevaluating Genetic Algorithm Performance Under Coordinate Rotation of Benchmark Functions." *BioSystems* 39: 263–278.

Salyers, A., and D. Whitt. 2001. *Microbiology: Diversity, Disease, and the Environment*. 1st ed. Bethesda, MD: Fitzgerald Science Press.

Shors, T. 2009. *Understanding Viruses*. 1st ed. Burlington, MA: Jones & Bartlett Learning.

Suganthan, P. N., N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. 2005. "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization." Technical Report, Nanyang Technological University, Singapore.

Tanegawa, S. 1983. "Somatic Generation of Antibody Diversity." *Nature* 302: 575–581.

Topley, W. C., S. Graham, S. Wilson, and F. Cox. 1998. *Topley and Wilson's Microbiology and Microbial Infections. Virology*. 9th ed. London: Hodder Arnold.

Woldemariam, K. M., and G. G. Yen. 2010. "Vaccine-Enhanced Artificial Immune System for Multimodal Function Optimization." *IEEE Transactions on Systems, Man, Cybernetics, Part B: Cybernetics* 40: 218–228.

Yang, Z., K. Tang, and X. Yao. 2008. "Large Scale Evolutionary Optimization Using Cooperative Coevolution." *Information Sciences* 178: 2985–2999.

Yao, X., Y. Liu, and G. Lin. 1999. "Evolutionary Programming Made Faster." *IEEE Transactions on Evolutionary Computation* 3: 82–102.