Applied Soft Computing 32 (2015) 72–79

Contents lists available at ScienceDirect

Applied Soft Computing

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

Ions motion algorithm for solving optimization problems

Behzad Javidy[a], Abdolreza Hatamlou[a,*], Seyedali Mirjalili[b]

[a] Department of Computer Science, Khoy Branch, Islamic Azad University, Khoy, Iran
[b] School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia

A R T I C L E   I N F O

Article history:
Received 24 June 2014
Received in revised form 18 March 2015
Accepted 19 March 2015
Available online 28 March 2015

Keywords:
Ions motion algorithm
Optimization problems
Meta-heuristic algorithms

A B S T R A C T

This paper proposes a novel optimization algorithm inspired by the ions motion in nature. In fact, the proposed algorithm mimics the attraction and repulsion of anions and cations to perform optimization. The proposed algorithm is designed in such a way to have the least tuning parameters, low computational complexity, fast convergence, and high local optima avoidance. The performance of this algorithm is benchmarked on 10 standard test functions and compared to four well-known algorithms in the literature. The results demonstrate that the proposed algorithm is able to show very competitive results and has merits in solving challenging optimization problems.

1. Introduction

Over the last two decades, numerous algorithms with inspiration from the nature have been proposed for solving various optimization problems. Some of the most popular are Genetic Algorithm (GA) [1–5], Differential Evolution Algorithm (DE) [6–10], Particle Swarm Optimization (PSO) [11–14], and Artificial Bee Colony (ABC) [15–19]. Optimization algorithms have advantages and disadvantages compared to each other and may show different performances when solving discrete and continuous problems.

There are two conflicting criteria when assessing two algorithms: convergence rate versus quality of the final solution. Fast convergence speed may result in premature convergence and entrapment in local optima. On the other hand, favoring quality of solutions may results in more extensive search of the search space and consequently slower convergence. To address these two issues, the researchers improve the current algorithms or propose new techniques. In this paper, a new optimization algorithm called Ions Motion Optimization (IMO) is proposed as a competitive algorithm in this field.

The IMO algorithm is a population-based algorithm inspired from properties of ions in nature. Our main objectives when designing this algorithm are to require IMO to have the least number of tuning parameters, low computational complexity, fast convergence, and high local optima avoidance. The main inspirations of the IMO algorithm are two different ions: anion (a negative charged particle) and cation (a positive charged particle). The IMO algorithm divides the population of candidate solutions to two sets of negative charged ions and positive charged ions, and improve them according to the important characteristics of the ions "*ions with the same charges repel each other, but with opposite charges attract each other*" [20].

In liquid state, the ions have greater freedom of motion compared to the solid phase (crystal) where high attraction forces between prevent ions from moving around freely. In fact, ions face minor motion and mostly vibrate in their position in solid phase. The IMO algorithm also mimics these two phases to perform diversification and intensification during optimization. The rest of the paper is organized as follows:

Section 2 provides the literature review of the recent meta-heuristics algorithms. Section 3 proposes the IMO algorithm. The results and discussion of the test functions are provided in Section 4. Eventually, Section 5 concludes the works and opens some avenues for future studies.

2. Literature review

Stochastic optimization techniques refer to the set of approaches that generate random solutions for an optimization problem. Based on the mechanism of the algorithm, the random solutions are combined in order to improve the oval quality of the initial solutions. This process is iterated until the satisfaction of a termination condition. A taxonomy of stochastic optimization algorithms here is based on the number of random solutions generated in each iteration. An algorithm may create single or multiple random solutions in every iteration.

* Corresponding author. Tel.: +98 9144627893.
E-mail address: hatamlou@iaukhoy.ac.ir (A. Hatamlou).

http://dx.doi.org/10.1016/j.asoc.2015.03.035
1568-4946/© 2015 Elsevier B.V. All rights reserved.

Simulated Annealing (SA) [21] is an example of the algorithms with single solution in every iteration. The optimization starts with one solution and it is improved over the course of iterations. The solution tends to randomly change its position based on a cooling factor. The higher cooling factor, the more sudden random changes in the solution. The cooling factor is increased over the iterations which results in convergence of the solution around an optimum. The limitation of such algorithm is that they are very likely to be trapped in local optima although the computational complexity is low.

Population-based algorithms belong to the stochastic optimization techniques with multiple solutions in each iteration. The optimization process starts with creating a set of random solutions (population). These solutions are then merged to create a new population. In order to guarantee improvement of the whole population, best solutions are usually selected for improving the quality of the solutions with poor quality. Obviously, the main advantage of these approaches is high local optima avoidance since a population is employed to search the search space. However, the computational complexity of the population-based algorithms is much higher than algorithms with single candidate solution.

The population-based algorithms themselves can be divided to three groups based on inspiration: swarm-inspired, evolution-inspired, and physics-inspired algorithms [22]. The swarm-based algorithms mostly mimic the social and individual behavior of swarm, herds, schools, or groups of creatures in nature. The PSO algorithm is the most popular swarm-inspired algorithm in this class, which imitate the collective behavior of birds. In this algorithm, candidate solutions are able to save and retrieve the best solutions they obtained so far as well as the best solution achieved by the whole swam. The convergence is guaranteed by moving toward the best positions obtained so far and the guidance by the best solution of the swarm.

Another well-known swarm-inspired algorithm is Ant Colony Optimization (ACO) proposed by Dorigo [23]. As its name suggests, this algorithm simulates the social and collective behavior of an ant colony. The main inspiration of this algorithm is the mechanism that ants utilize pheromone to find the shortest path from nest to foods. At every iteration, search agents of ACO record the history of solutions and qualities in order to fill out a pheromone matrix and eventually improve other solutions. The Artificial Bee Colony (ABC) [15–19] is a similar method that mimics the social life style of bees in a bee colony. In this algorithm, the search agents are divided into different groups to explore (scout bees) and exploit (onlooker and employed) the search space. Some of the other algorithms in this class are Bat Algorithm proposed by Yang [24], Glowworm Swarm Optimization (GSO) proposed by Krishnanand and Ghose [25], Multi-swarm optimization [26], Gray Wolf Optimizer [22], Artificial Fish-Swarm Algorithm (AFSA) [27], Firefly Algorithm (FA) [28], Cuckoo Search (CS) [29], Krill Herd (KH) [30], and Heart Algorithm [31].

The second class of algorithms is evolution-inspired algorithm. Such algorithms mostly simulate evolutionary phenomena in nature. Similar to other population-based algorithm, the optimization process starts with a set of random solutions. Then, three main operators evolve the initial population: selection, re-production, and mutation. The selection operator is responsible for choosing proper individuals based on their fitness values. The re-production operator combines the selected individuals by the selection operator. Eventually, the mutation operator randomly changes the re-produced new individuals in order to maintain diversity of the whole population. The most well-known algorithm in this class is GA [32]. This algorithm considers candidate solutions as chromosomes and the parameters as genes. In every generation, the chromosomes with higher fitness values have higher chance to crossover with other chromosomes. Therefore, the overall fitness of all chromosomes is increased over the course of iterations. Some of the other algorithms in this class are Differential Evolution (DE) [33], Evolution Strategy (ES) [34], Genetic Programming (GP) [35], and Biogeography-based Optimizer (BBO) [36].

The last class of algorithms is physics-based algorithms where the main inspiration mostly originates from physical rules and phenomena in nature. Similar to the other two classes, optimization is done by a group of candidate solutions called search agents. The key difference here is that the search agents are moved/combined based on physics-inspired concepts. For instance, Magnetic Optimization Algorithm (MOA) [36] simulates the electromagnetic forces between electromagnetic particles to move the search agents around the search space. Since the electromagnetic force is proportional to the fitness of particles, search agents tend to be attracted toward the fittest particles. Therefore, the search agents of this algorithm are improved by moving toward the best solutions. A similar algorithm to MOA is Gravitational Search Algorithm (GSA) [37]. The GSA algorithm considers the search agent as masses that attract each other based on the gravitational forces between them, which are again proportional to their fitness functions. Regarding to the movement of masses, Newtonian law of motion is also utilized by the GSA algorithm. Some of the other algorithms in this class are Ray Optimization (RO) [38], States of Matter Search (SMS) [39], Big-Bang Big-Crunch (BBBC) [40], Black Hole (BH) [41], Artificial Chemical Reaction Optimization Algorithm (ACROA) [42], and Kinetic Gas Molecules Optimizer [43].

All the algorithms in three classes have their own advantages and disadvantages. According to no-free-lunch theorem [44], none of them is able to solve all optimization problems. Regardless of differences in the mechanisms of population-based algorithms in this field, the common is the division of the search process to two main milestones: diversification versus intensification. Diversification refers to the milestone where candidate solutions tend to be merged more frequently and find promising areas of the search space. In other words, candidate solutions face sudden changes in diversification milestone in order to explore the search space as broad as possible. Contradictory, candidate solutions are prone to very little changes in the intensification milestone. In fact, intensification milestone promotes convergence toward the best solutions obtained in the diversification milestone. As discussed in Section 1, these two phases are in conflict. Favoring diversification result in higher local optima avoidance, whereas emphasizing intensification yields to faster convergence rate. The following section proposes a new physics-based algorithm with two specific milestones for diversification and intensification.

3. Ions motion optimization (IMO) algorithm

This section first discusses the inspirations of the IMO algorithm. The mathematical model and the algorithm are then presented.

3.1. Inspirations

The word "*ion*" is a Greek term. English physician Michael Faraday introduced this term in 1834. Generally speaking, charged particles are called ion and can be divided to two types:

- Anion: ions with negative (−) charge.
- Cation: ions with positive (+) charge.

The conceptual model of anions and cations are illustrated in Fig. 1.

The main inspiration of the IMO algorithm is the fact that ions with similar charges tend to repel, whereas ions with opposite

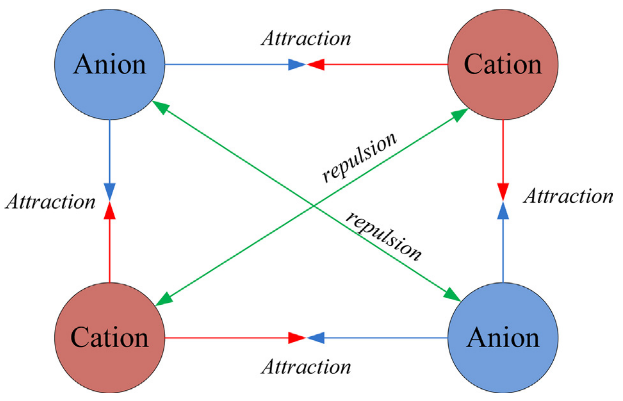

Fig. 1. Conceptual models and anion, cation, attraction force, and repulsion force.

charges attract each other [20]. The attraction and repulsion forces between anions and cations are also depicted in Fig. 1.

The candidate solutions for a given optimization problem in IMO algorithm are divided into two groups anions (negative ions) and cations (positive ions). The ions represent candidate solutions for a particular problem and attraction/repulsion forces move the ions around the search space. We require IMO algorithm's ions to move toward best ions with opposite charges. The ions are evaluated based on their fitness, so the fitness of ions is proportional to the value of the objective function. Needless to say, anions move toward the best cation, whereas cations move toward the best anion. Their amount of movement depends on the attraction/repulsion forces between them. The size of this force specifies momentum of each ion.

So far, the movement of ions in IMO algorithm can guarantee the improvement of all ions over the course of iterations. However, there are no mechanisms for diversification and intensification. In order to require the ions to diversify and intensify, we assume that the ions can be in two completely different phases: liquid versus solid.

3.2. Liquid phase (diversification)

Ions in liquid have more freedom to move. In addition, attraction forces among ions with opposite charges are more than repulsion force among ions with similar charges [20]. We require IMO to ignore repulsion forces in this phase in order to explore the search space. We assume that the only factor for computing attraction force is the distance between ions, so the mathematical model is proposed as follows:

$$AF_{i,j} = \frac{1}{1+e^{-0.1/AD_{i,j}}} \tag{3.1}$$

$$CF_{i,j} = \frac{1}{1+e^{-0.1/CD_{i,j}}} \tag{3.2}$$

where $AD_{i,j} = \left|A_{i,j} - Cbest_j\right|$, $CD_{i,j} = \left|D_{i,j} - Abest_j\right|$, i is the ion index, j indicates dimension, $AD_{i,j}$ is the distance of i-th anion from the best cation in j-th dimension, $CD_{i,j}$ calculates the distance of i-th cation from the best anion in j-th dimension, $AF_{i,j}$ is the resultant attraction force of anions, and $CF_{i,j}$ indicates the resultant attraction force of cations.

According to Eqs. (3.1) and (3.2), forces between ions are inversely proportional to the distances between them. The greater distance, the lower attraction force. In other words, when the ions' distance becomes greater from the best ion with opposite charge, a lower attraction force is applied to them. According to Eq. (3.1), the attraction force varies between 0.5 and 1.

After calculating the force, the position of anions and cations are updated as follows:

$$A_{i,j} = A_{i,j} + AF_{i,j} \times (Cbest_j - A_{i,j}) \tag{3.3}$$

$$C_{i,j} = C_{i,j} + CF_{i,j} \times (Abest_j - C_{i,j}) \tag{3.4}$$

where i is the ion index and j indicates the dimension index.

According to the Eqs. (3.3) and (3.4), only force element determines momentum of each ion toward the best ion with opposite charge. $Cbest_j$ and $Abest_j$ indicates the best cation and anion respectively.

As may be inferred from the mathematical models, there is no random component involved liquid phase. Since the movements are proportional to distances, however, the ions diversity around the search space. Fig. 2 shows a conceptual model of ions motion in the liquid phase.

Since ions tend to be attracted to other ions with opposite charges, exploration can be guaranteed. With increasing the number of iterations, more ions interact and converge toward the best ions with opposite charges and the diversity/exploration are gradually decreased. This is what exactly happens in nature when ions re-form a crystal from a liquid. The search agents of IMO also enter the crystal phase, meaning that they eventually converge toward a solution in the search space.

3.3. Crystal phase (intensification)

In crystal phase, ions have been converged at an optimal point and convergence has occurred. Due to the unknown shape of search spaces, however, the convergence could be occurred toward local optima. Therefore, we propose a mechanism to exit the trapped ions from local optima in case of local optima entrapment in this phase.

In nature, the anions and cations in ionic crystal are arranged to maximize their attractions. When an external force is applied to the same charges in the solid phase, the produced repulsions force crack the solid apart [20]. We mathematically model this phenomenon in order to resolve local optima entrapment as follows:

```
if ($CbestFit >= CworstFit/2$ and $AbestFit >= AworstFit/2$)
  if $rand() > 0.5$
    $A_i = A_i + \Phi_1 \times (Cbest - 1)$
  else
    $A_i = A_i + \Phi_1 \times (Cbest)$
  end if
  if $rand() > 0.5$
    $C_i = C_i + \Phi_2 \times (Abest - 1)$                    (3.5)
  else
    $C_i = C_i + \Phi_2 \times (Abest)$
  end if
  if $rand() < 0.05$
    $Re-initialized\ A_i\ and\ C_i$
  end if
end if
```

where Φ_1 and Φ_2 are random numbers in $[-1, 1]$ and $rand()$ is a function that returns a random number in $[0, 1]$.

According to above pseudo codes, there is a condition for transiting from liquid state to solid state where the average fitness of worst ions should be equal or smaller than the best ions' fitness. If this condition satisfied, anions and cations are randomly re-spanned around the best cation and anion respectively. Please

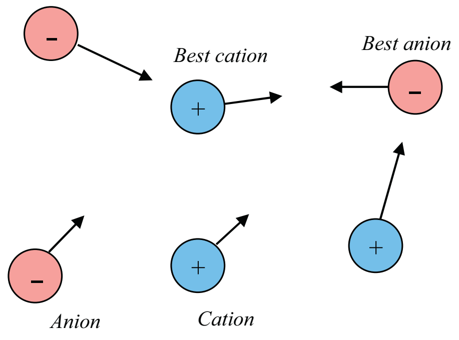

Fig. 2. Ions motion toward best ions in liquid phase.

note that the parameter Φ is a random number between 1 and -1. The parameters *CworstFit* and *CbestFit* are the fitness of the worst and the best cation respectively. In addition, *AworstFit* and *AbestFit* are the fitness of the worst and the best anion. In order to increase the diversity, eventually, a number of anions and cations are re-initialized with low probability. It also should be noticed that although it was possible to merge the first two if-else statements in (3.5), we deliberately split these two conditions to provide the following four cases:

- Both first if-else statements were satisfied:

$$A_i = A_i + \Phi_1 \times (Cbest - 1)$$
$$C_i = C_i + \Phi_2 \times (Abest - 1)$$

- The first if-else statement was satisfied but the second one was not:

$$A_i = A_i + \Phi_1 \times (Cbest - 1)$$
$$C_i = C_i + \Phi_2 \times (Abest)$$

- The first if-else statement was not satisfied but the second one was:

$$A_i = A_i + \Phi_1 \times (Cbest)$$
$$C_i = C_i + \Phi_2 \times (Abest - 1)$$

- Both first if-else statements were not satisfied:

$$A_i = A_i + \Phi_1 \times (Cbest)$$
$$C_i = C_i + \Phi_2 \times (Abest)$$

In contrast, combining the first two if-else statements results in only two possible situations:

- Combined if-else statements was satisfied:

$$A_i = A_i + \Phi_1 \times (Cbest - 1)$$
$$C_i = C_i + \Phi_2 \times (Abest - 1)$$

- Combined if-else statements was not satisfied:

$$A_i = A_i + \Phi_1 \times (Cbest)$$
$$C_i = C_i + \Phi_2 \times (Abest)$$

It may see that splitting these two conditions provide a diverse behavior for the proposed algorithm, which is fruitful for resolving local optima stagnations.

Fig. 3 illustrates the distribution of ions in crystal phase. This phase maintains the diversity of ions during optimization. After this

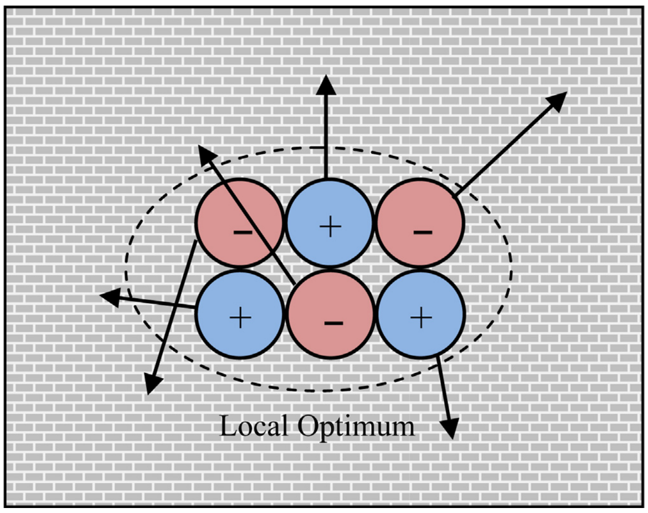

Fig. 3. Ions motion around an optimum in the crystal phase.

phase, the ions again enter the liquid phase and star exploration of the search space. The general steps of the IMO algorithm are presented in Fig. 4.

It may be seen in Fig. 4 that the IMO algorithm begins optimization with creating a set of random solutions. Note that all the random solutions at the beginning or during optimization are generated using $r(ub_i - lb_i) + lb_i$ equation where r is random number generated with uniform distribution in the interval [0, 1], lb_i shows the lower bound of the ith variable, and ub_i indicates the upper bound of the ith variable. At this stage, the ions are randomly and equally divided into two groups: anions and cations. Then, the

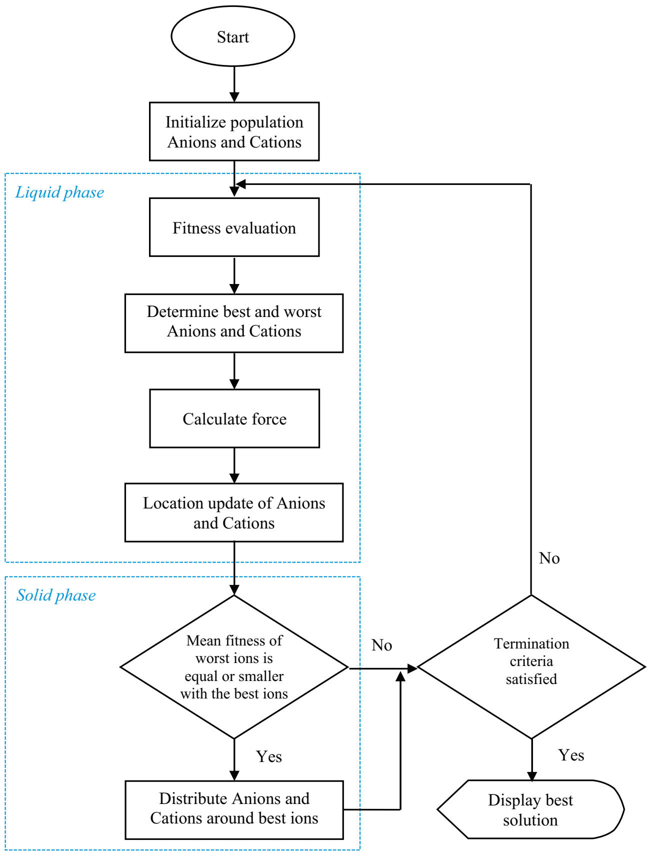
Start
Initialize population Anions and Cations
Liquid phase
Fitness evaluation
Determine best and worst Anions and Cations
Calculate force
Location update of Anions and Cations
Solid phase
Mean fitness of worst ions is equal or smaller with the best ions
No
Yes
Distribute Anions and Cations around best ions
Termination criteria satisfied
No
Yes
Display best solution

Fig. 4. General steps of the IMO algorithm.

corresponding fitness of all ions are calculated. According to calculated fitness values, the best and worst anions are chosen and saved in proper variables. The attraction forces and position are updated by Eqs (3.1), (3.2), (3.3) and (3.4) respectively. At every iteration, if the condition of crystal phase is met, the ions go the crystal phase. The ions keep going to liquid and solid phases until the satisfaction of an end criterion. Eventually, the best ion is reported as the best approximation of the global optimum.

It may be inferred from the optimization steps that the operation of the IMO algorithm starts with creating a set of random solutions for a given problem. The problem here can be a test problem or real problem. The IMO algorithm keeps changing the parameters of the problem and monitors the outputs. The main operation steps of the IMO algorithm occurs when deciding about the way of combining the solutions with respect their corresponding calculated outputs by the problem. The flowchart in Fig. 4 clearly shows that the modification of solutions is done in two phases discussed in details above.

It is worth mentioning here that the problem representation for the proposed algorithm is very simple and straightforward. We designed the IMO algorithm in such a way to accept all the inputs as a single vector. Therefore, any optimization problem can be formulated as a minimization problem (without loss of generality) for the proposed IMO algorithm as follows:

$$Minimize: \quad F(\vec{X}) \tag{3.6}$$

$$Subject\ to: \quad lb_i \le x_i \le ub_i \quad i = 1, 2, \ldots, n \tag{3.7}$$

where $\vec{X} = [x_1, \ldots, x_n]$, F is the objective function, n is number of parameters (variables of the problem), lb_i shows the lower bound of the ith variable, and ub_i indicates the upper bound of the ith variable.

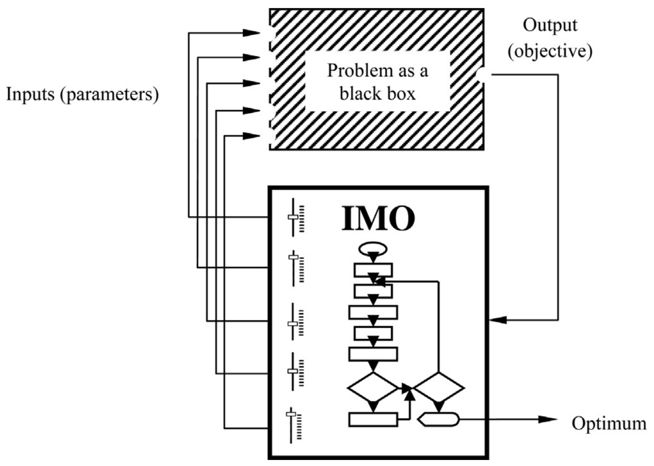

Fig. 5. IMO considers optimization problems as black boxes and finds their optimum.

As may be inferred from Eqs. (3.6) and (3.7) that problem representation is independent of the structure of the proposed algorithm. In other words, a designer does not necessarily have to know about the structure of the IMO algorithm, yet the problem formulation is the key step for utilizing the IMO algorithm. Once the problem is formulated by identifying and defining the parameters, range of parameters, and objective function, a designer is able to employ the IMO algorithm to determine the global optimum of the problem.

The independency of the problem and IMO is due to considering problem as black boxes as shown in Fig. 5. This means that this algorithm only provides a set of solutions for the given problem and observes the corresponding outputs. Based on the inputs and

Table 1
Benchmark functions used in experiments.

F	Function	Function definition	Range	Optimum
f_1	Sphere	$f(x)=\sum_{i=1}^{d} x_i^2$	[−100, 100]	0
f_2	Rosenbrock	$f(x)=\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	[−30, 30]	0
f_3	Schwefel	$f(x)=\sum_{i=1}^{d}\lvert x_i\rvert+\prod_{i=1}^{d}\lvert x_i\rvert$	[−10, 10]	0
f_4	Step	$f(x)=\sum_{i=1}^{d}\left(\lfloor x_i+0.5\rfloor\right)^2$	[−100, 100]	0
f_5	Zakharov	$f(x)=\sum_{i=1}^{d}x_i^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^4$	[−5, 10]	0
f_6	Powell	$f(x)=\sum_{i=1}^{d/4}[(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-2x_{4i-1})^4+10(x_{4i-3}+x_{4i})^4]$	[−4, 5]	0
f_7	Griewank	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[−600, 600]	0
f_8	Ackley	$f(x)=-a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-\exp\left(\frac{1}{d}\sqrt{\sum_{i=1}^{d}\cos(cx_i)}\right)+a+\exp(1)$	[−32, 32]	0
f_9	Rastrigin	$f(x)=10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$	[−5.12, 5.12]	0
f_{10}	Levy	$f(x)=\sin^2(\pi w_1)+\sum_{i=1}^{d-1}(w_i-1)^2[1+10\sin^2(\pi w_i+1)]+(w_d-1)^2[1+\sin^2(2\pi w_d)]$, *where* $w_i=1+\frac{x_i-1}{4}$, *for all* $i=1,\ldots,d$	[−10, 10]	0

Table 2
The initial parameters of algorithms.

Algorithm	Tuning parameter	Value
PSO [6]	Topology	Fully connected
	Cognitive constant (C_1)	1.8
	Social constant (C_2)	1.8
	Inertia constant (w)	0.6
GA [16]	Type	Real coded
	Selection	Roulette wheel
	Crossover	Single point (probability = 0.8)
	Mutation	Uniform (probability = 0.01)
DE [46]	Differential variation	0.5
	Crossover rate	0.9
ABC [16]	Limit	SN^*D

outputs, the proposed algorithm makes decision as to how to change the inputs to improve the change of improving the candidate solutions and consequently obtain better output.

In order to better understand the behavior of the proposed IMO algorithm, some comments on the novel concepts and ideas as well as their effects on the performance of the proposed algorithms are as follows:

- The proposed method of interaction between the search agents (ions) of the IMO algorithm using attraction and repulsion forces are novel, which assist the search agent to move around the search space and explore diverse regions of search spaces.
- Eqs. (3.1) and (3.2) have been adapted in the field of evolutionary optimization for the first time, which assist the search agents of IMO to update their positions with respect to the best solutions obtained so far.
- Conditions in (3.5) allow search agents to change their locations randomly and abruptly without considering attraction and repulsion forces applied. This mechanism is useful for resolving local optima stagnation during optimization.
- The proposed equations for simulating liquid phase cause convergence of search agents toward the recent best solutions found so far. This approach guarantees the convergence of the IMO algorithm toward the best solutions during optimization.
- The proposed equations and conditions for simulating the crystal phase are useful for resolving local optima stagnation. The liquid phase may cause the search agents to be trapped in local solutions quickly. However, the crystal phase resolves such cases by relocating the search agents around the search space suddenly and stochastically.

4. Results and discussion

To examine the performance of optimization algorithms, benchmarks functions are usually employed. In this paper, 10 test functions are employed to examine the efficiency and performance

Table 4
Mean run time obtained by ABC and IMO algorithm.

F	Alg.	ABC	IMO
	Dim	Time (s)	
f_1	30	3.2905	0.3664
f_2	30	27.884	0.4783
f_3	30	6.3781	0.4225
f_4	30	1.7662	0.2178
f_5	10	48.5488	0.4081
f_6	24	34.3237	0.5764
f_7	30	6.0277	0.3751
f_8	30	6.7684	0.3651
f_9	30	5.1436	0.3279
f_{10}	30	5.2936	0.4078

Table 5
Results obtained by IMO algorithm.

F	Dim	Best-v	Mean-v	SD	Iter	Time (s)
f_1	500	0	0	0	171	1.764
	1000	0	0	0	175	3.1584
	5000	0	0	0	180	18.512
f_2	500	0	0	0	144	1.54
	1000	0	0	0	151	2.8715
	5000	0	0	0	149	16.1764
f_3	500	0	0	0	147	1.4664
	1000	0	0	0	144	2.5782
	5000	0	0	0	145	15.1603
f_4	500	0	0	0	12	0.3366
	1000	0	0	0	13	0.4553
	5000	0	0	0	12	1.7235
f_5	500	0	0	0	170	1.8483
	1000	0	0	0	174	3.3797
	5000	0	0	0	168	18.2349
f_6	500	0	0	0	122	2.9622
	1000	0	0	0	125	5.7987
	5000	0	0	0	126	28.3783
f_7	500	0	0	0	55	0.86
	1000	0	0	0	61	1.5755
	5000	0	0	0	65	8.721
f_8	500	8.88E−16	8.88E−16	2.07883E−31	1000	8.9322
	1000	8.88E−16	8.88E−16	2.07883E−31	1000	17.5463
	5000	8.88E−16	8.88E−16	2.07883E−31	1000	105.1334
f_9	500	0	0	0	48	0.6778
	1000	0	0	0	54	1.2031
	5000	0	0	0	52	5.8387
f_{10}	500	1.50E−32	1.50E−32	2.88496E−48	1000	25.1106
	1000	1.50E−32	1.50E−32	2.88496E−48	1000	49.298
	5000	1.50E−32	1.50E−32	2.88496E−48	1000	239.9475

Population size was 50 and maximum iteration was 1000 for each run that runs were repeated 30 times per test function.

Table 3
Results obtained by GA PSO, ABC, and IMO algorithms.

F	Alg.	GA		PSO		DE		ABC		IMO	
	Dim	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_1	30	1.11E+03	7.42E+01	0	0	0	0	0	0	0	0
f_2	30	1.96E+05	3.85E+04	1.51E+01	2.42E+01	1.82E+01	5.04E+00	8.88E−02	7.74E−02	0	0
f_3	30	1.10E+01	1.39E+00	0	0	0	0	0	0	0	0
f_4	30	1.17E+03	7.66E+01	0	0	0	0	0	0	0	0
f_5	10	1.34E−02	4.53E−03	0	0	0	0	2.48E−04	1.83E−04	0	0
f_6	24	9.70E+00	1.55E+00	1.10E−04	1.60E−04	2.17E−07	1.36E−7	3.13E−03	5.03E−04	0	0
f_7	30	1.06E+01	1.16E+00	1.74E−02	2.08E−02	1.48E−03	2.96E−03	0	0	0	0
f_8	30	1.47E+01	1.78E−01	1.65E−01	4.94E−01	0	0	0	0	0	0
f_9	30	5.29E+01	4.56E+00	4.40E+01	1.17E+01	1.17E+01	2.54E+00	0	0	0	0
f_{10}	30	N/A	N/A	N/A	N/A	N/A	N/A	0	0	0	0

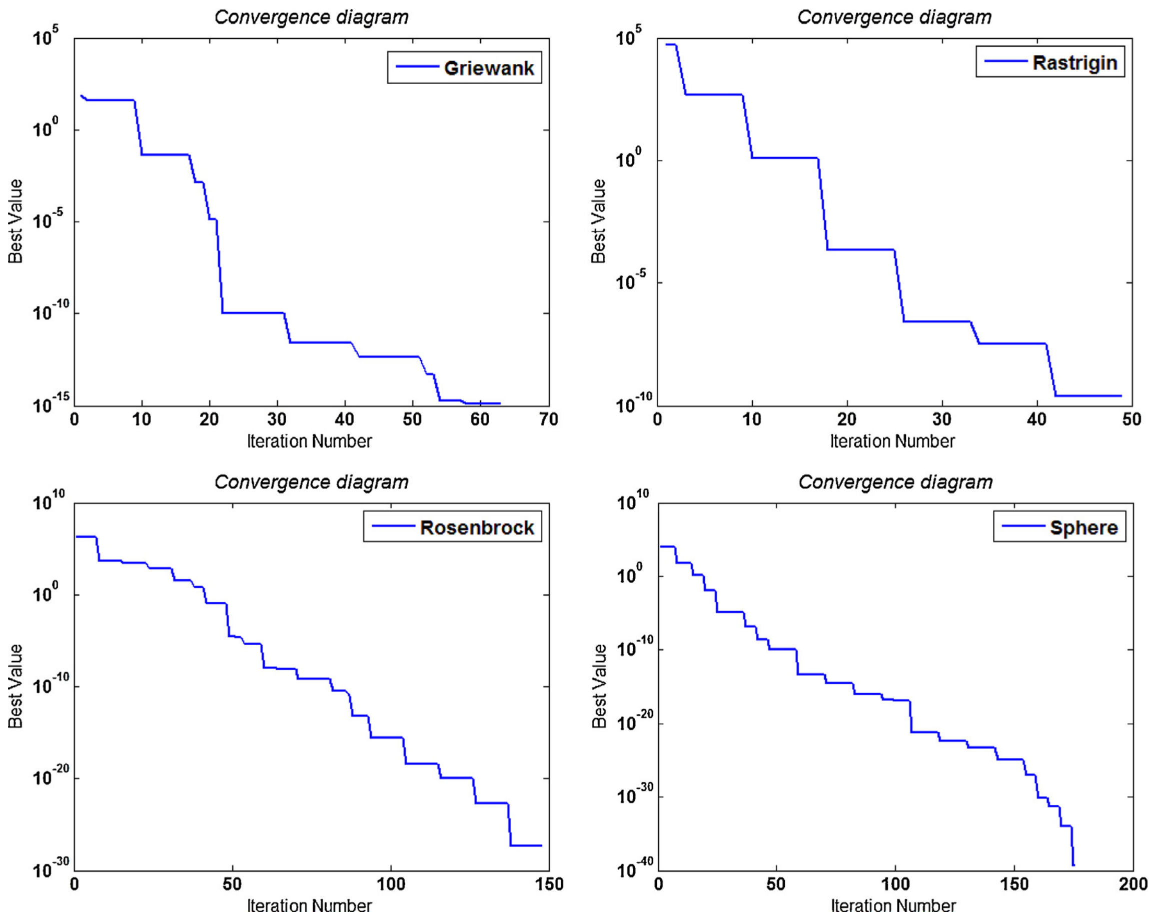

Fig. 6. Convergence diagrams.

of the IMO algorithm. The mathematical formulation, domain of parameters, and optima are presented in Table 1. The functions of f_1 to f_6 are unimodal, whereas f_7 to f_{10} are multimodal. Unimodal functions have only one local minimum, while multimodal functions have multiple local minima [45]. We compare the performance of the proposed algorithm with other algorithms in Table 3. The comparison criteria are average and standard deviation of the best solutions obtained over 30 independent runs. We also compare the runtime of the IMO algorithm with ABC algorithm in Table 4.

4.1. Experiments settings

The main tuning parameters of the algorithms are population size and the maximum number of iterations. In this work, we set the size of the population to 50 for all algorithms. The maximum numbers of iterations are also equal to 10,000 for GA, PSO, DE, and ABC, and 1000 for IMO. Other initial parameters of the algorithms are provided in Table 2.

We solve each test function by the algorithms 30 times and report the statistical results in Tables 3 and 4.

4.2. Results

As may be seen in Table 3, the IMO algorithm is compared to PSO, DE, ABC, and GA. Please note that the value less than 2^{-10} is considered as 0. Table 3 shows that the IMO has the best mean and standard deviation among other algorithms. The convergence of the IMO algorithm toward the optima during 30 runs is quite evident according to the results. Meanwhile, the maximum iteration is 1000 for IMO algorithm and 10,000 for the other algorithms, evidencing the fast convergence for this algorithm.

Table 4 provides average runtime of the IMO algorithm compared with ABC algorithm. It should be noted that the end criterion in this experiment is considered as the optimum value less than 10^{-12}. The results show the average run time of IMO algorithm is significantly lower than that of ABC algorithm.

Other experimental results in this section are provided in Table 5. We increase the dimension of the test functions in order to benchmark the performance of the IMO algorithm when solving extremely high-dimensional problems. The considered dimensions are 500, 1000, and 5000 dimensions. The results show that IMO shows very good performance and fast runtime for high dimensional problems as well. To further investigate convergence behavior of this algorithm, the convergence curves of the IMO algorithm when solving high-dimensional problems are also illustrated in Fig. 6. The convergence curves show that IMO show steady convergence over the course of iterations.

To sum up, the results show that the proposed algorithm is able to show highly superior performance on test functions. The results of unimodal test functions proved that this algorithm has very fast convergence rate. This is due to the proposed liquid phase where the ions tend to be attracted toward the best anion and cation. This results in quick convergence toward promising regions of the search spaces. The results of the multimodal test functions also evidence superior local optima avoidance of the IMO algorithm. This is because of the crystal phase where the ions are obliged to exit from the local optima by the repulsion forces.

5. Conclusion

In this paper a novel population-based meta-heuristic inspired from ions was proposed. Two phases were designed in order to explore and exploit the search space. The ions in IMO algorithm were required to continuously transit from liquid phase to crystal phase to find the optimum. Ten standard test functions as well as three real problems were employed to benchmark the performance of the proposed method. The results showed that IMO provides highly competitive results compared to the well-known algorithms in this field. The findings showed that the crystal phase assists IMO to resolve local optima entrapment. In addition, the liquid phase can guarantee convergence of the ions toward an optimum in the search space. The proposed algorithm also were not equipped by any tuning parameters and designed in a way to automatically adapt itself to the search spaces. According to this comprehensive study, we offer this algorithm to the researchers in this field and other fields. For future work, we recommend improving the performance of the proposed algorithm using chaotic maps, evolutionary operators such as mutation and crossover, and hybridizing with other algorithms. Another research direction is to modify the proposed algorithm to be able to handle multiple objectives for solving multi-objective problems.

References

[1] D. Beasley, D. Bull, R. Martin, An Overview of Genetic Algorithms. Part 2, University of Cardiff, Cardiff, 1993.
[2] D. Beasley, D. Bull, R. Martin, An Overview of Genetic Algorithms. Part 1, University of Cardiff, Cardiff, 1993.
[3] D.B. Fogel, What is evolutionary computation, in: IEEE Spectrum, 2000, pp. 26–32.
[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading, 1989.
[5] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1996.
[6] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation (CEC' 2004), Piscataway, NJ, 2004, pp. 1980–1987.
[7] K. Price, R. Storn, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[8] K. Price, R. Storn, Differential Evolution a Practical Approach to Global Optimization, Springer Natural Computing Series, 2005.
[9] P. Bergey, C. Ragsdale, Modified differential evolution: a greedy random strategy for genetic recombination, Omega 33 (2005) 255–265.
[10] A. Salman, A. Engelbrecht, M. Omran, Empirical analysis of self-adaptive differential evolution, Eur. J. Oper. Res. 183 (2007) 785–804.
[11] J. Kennedy, R.C. Eberhart, A Discrete Binary Version of the Particle Swarm Algorithm, 1997, pp. 4104–4108.
[12] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, 1995, pp. 1942–1948.
[13] M. Clerc, J. Kennedy, The particle swarm: explosion, stability and convergence in multi-dimensional complex space, IEEE Trans. Evol. Comput. 20 (2002) 1671–1676.
[14] O.M. Nezami, A. Bahrampour, P. Jamshidlou, Dynamic Diversity Enhancement in Particle Swarm Optimization (DDEPSO) algorithm for preventing from premature convergence, Procedia Comput. Sci. 24 (2013) 54–65.
[15] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, in: Information Sciences, 2012, pp. 120–142.
[16] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (2009) 108–132.
[17] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, in: Applied Soft Computing, 2008, pp. 687–697.
[18] D. Karaboga, An idea based on honey bee swarm for numerical optimization, in: Technical Report-TR06, 2005.
[19] W. Gu, M. Yin, C. Wang, Self adaptive artificial bee colony for global numerical optimization, IERI Procedia 1 (2012) 59–65.
[20] M.S. Silberberg, Principles of General Chemistry, McGraw-Hill, New York, 2007.
[21] P.J. Van Laarhoven, E.H. Aarts, Simulated Annealing, Springer, 1987.
[22] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[23] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (2006) 28–39.
[24] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, 2010, pp. 65–74.
[25] K. Krishnanand, D. Ghose, Glowworm swarm optimisation: a new method for optimising multi-modal functions, Int. J. Comput. Intell. Stud. 1 (2009) 93–119.
[26] B. Niu, Y. Zhu, X. He, H. Wu, O. MCPS, A multi-swarm cooperative particle swarm optimizer, Appl. Math. Comput. 185 (2007) 1050–1062.
[27] X.-L. Li, J.-X. Qian, Studies on artificial fish swarm optimization algorithm based on decomposition and coordination techniques, J. Circuits Syst. 1 (2003) 1–6.
[28] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspir. Comput. 2 (2010) 78–84.
[29] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: Nature & Biologically Inspired Computing, NaBIC 2009, World Congress on IEEE, 2009, pp. 210–214.
[30] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (2012) 4831–4845.
[31] A. Hatamlou, Heart: a novel optimization algorithm for cluster analysis, Prog. Artif. Intell. 2 (2014) 167–173.
[32] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–72.
[33] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[34] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[35] J.R. Koza, Genetic Programming, 1992.
[36] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[37] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[38] A. Kaveh, M. Khayatazad, A new meta-heuristic method: ray optimization, Comput. Struct. 112 (2012) 283–294.
[39] E. Cuevas, A. Echavarría, M.A. Ramírez-Ortegón, An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation, Appl. Intell. 40 (2014) 256–272.
[40] O.K. Erol, I. Eksin, A new optimization method: big bang–big crunch, Adv. Eng. Softw. 37 (2006) 106–111.
[41] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. 222 (2013) 175–184.
[42] B. Alatas, ACROA: artificial chemical reaction optimization algorithm for global optimization, Expert Syst. Appl. 38 (2011) 13170–13180.
[43] S. Moein, R. Logeswaran, KGMO. A swarm optimization algorithm based on the kinetic energy of gas molecules, Inf. Sci. 275 (2014) 127–144.
[44] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[45] M. Jamil, X. Yang, A literature survey of benchmark functions forglobal optimization problems, Int. J Math. Modell. Numer. Optim. 4 (2013) 150–194.
[46] D. Corne, M. Dorigo, F. Glover, New Ideas in Optimization, McGraw-Hill, 1999.