Communications in Nonlinear Science and Numerical Simulation 125 (2023) 107338

Contents lists available at ScienceDirect

Communications in Nonlinear Science and Numerical Simulation

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

Research paper

Leopard seal optimization (LSO): A natural inspired meta-heuristic algorithm

Asmaa H. Rabie *, Nehal A. Mansour, Ahmed I. Saleh

Computer Engineering and Systems Department, Faculty of engineering, Mansoura University, Mansoura, Egypt

ARTICLE INFO

Article history:
Received 13 November 2022
Received in revised form 14 May 2023
Accepted 31 May 2023
Available online 3 June 2023

Keywords:
Exploration
Exploitation
Leopard seal
Natural inspired
Optimization

ABSTRACT

The main objective of this paper is to introduce a new NIO algorithm inspired from the hunting strategy of the leopard seals called Leopard Seal Optimization (LSO)to provide a simple swarm intelligence algorithm that have a high flexibility to solve real-time engineering problems in a fast and more accurate manner without falling into local optima problem. LSO is compared against recent NIO algorithms considering feature selection for disease diagnosing as the underlying optimization problem. In experimental results, LSO has been statistically tested against a recent six swarm intelligence algorithms using Wilcoxon test and t-test with significance level equals 0.05 based on the common five benchmark functions. These recent algorithms are called Tuna Swarm Optimization (TSO), Pelican Optimization Algorithm (POA), Cat and Mouse-Based Optimization (CMBO), Aphid–Ant Mutualism (AAM), White Shark Optimizer (WSO), and Red Piranha Optimization (RPO). The statistical analysis proved that LSO outperforms other recent techniques in most cases where the most tested results are less than 0.05. Then, LSO has been tested against the same algorithms in binary version as feature selection algorithms using two metrics called accuracy and execution time. It is noted that LSO is faster and more accurate than other algorithms to determine the optimal set of features at all numbers of search agents. Finally, it is concluded that LSO outperforms other techniques using accuracy, execution time, Wilcoxon test, and t-test metrics. Compared to the recent techniques using the maximum iteration number=400, LSO provided the maximum accuracy value that equals 97.62% at the search agents number=120 and the minimum execution time that equals 1314 at the search agents number= 40.

1. Introduction

Natural Inspired Optimization (NIO) algorithms have achieved great success, especially in recent years, and the reason for this is their high flexibility and capability in solving the NP-hard problems subjected to complex constraints [1–3]. Several NIO have been introduced taking the inspiration from wolves, birds, fish, insects, ant, and lion etc., which are the amazing creatures of nature. In addition to its high efficiency, NIO algorithms are fast and easy to implement, as it needs a few algorithm parameters. Hence, they are applicable in solving wide range of complex engineering problems [1–8]. However, solving a problem using a NIO algorithm might be confusing due to the large number of proposals that exist today. Moreover, not every algorithm is suitable for all types of problems. Depending on the nature of the problem

* Corresponding author.
E-mail address: asmaa91hamdy@yahoo.com (A.H. Rabie).

https://doi.org/10.1016/j.cnsns.2023.107338
1007-5704/

to be solved, some are better than others. Therefore, there is an urgent need to devise new optimization problems inspired by nature that can solve and cover a wide area of problems in addition to their efficiency, speed and ease of implementation at the same time. There are two basic abilities for any meta-heuristic algorithm which are exploration and exploitation [2,9,10]. Exploration refers to searching for discovering new regions of the search space while exploitation refers to the search in close vicinity to previously discovered regions [2,10].

In fact, there are many different optimization algorithms provided to solve real time issues and to give the best solutions. These algorithms were categorized into nine different classes called physics, social, music, swarm, chemistry, biology, sports, mathematics, and hybrid algorithms [11–14]. These algorithms such as special relativity search as physics algorithm [11,12], social network search as social algorithm [11,12], stochastic paint optimizer as music algorithm [11,12], particle swarm optimizer as swarm algorithm [15], and atomic orbital search as chemistry algorithm [11,12]. Additionally, genetic algorithm is a biology algorithm [11,16], tiki-taka algorithm is a sport algorithm [17], chaos game optimization is a mathematics algorithm [11,12], and colonial competitive differential evolution is a hybrid algorithm that can be categorized as both biology and social algorithms [11]. Nowadays, there are new optimization algorithm called water-based algorithm such as groundwater flow algorithm that simulated the movement of groundwater to be applicable to many different problems [18]. Also, there is an improvement version of grey wolf optimization as a swarm intelligence algorithm called group-based synchronous–asynchronous grey wolf optimization [19]. This improved version can overcome the problems of original GWO by using synchronous–asynchronous approach. Although there are several optimization algorithms used to provide optimal solutions to real time problems, researchers are still looking for other new optimization algorithms that can outperform other algorithms by optimally solving problems and providing minimal execution time

In the Antarctic, the leopard seal rules the animal kingdom. This top predator has smart hunting behaviors and yet, it has not been deeply studied. Few scientific studies have focused on leopard seals; however, it is generally known that they are inquisitive, playful, and smart, creatures. Although Leopard seals are known to be solitary creatures that live and hunt alone and often compete with each other for food, they can sometimes engage in cooperative feeding. Hence, a number of individuals can cooperate to forage and hunt potential preys. Despite their solitary nature, leopard seals are quite vocal creatures as they have several types of calls which include barking, trembling, and whining. These vocal calls help them attract potential mates as well as establish areas of dominance. Hence, they have unique behaviors associated with each of these calls.

In this paper, the main contribution is centralized in introducing a new NIO algorithm that simulates the attractive attack behavior of the leopard seals, thus, it is called Leopard Seal Optimization (LSO) algorithm. LSO represents an attempt to derive an applicable and easy natural inspired optimization algorithm that can solve complex optimization problems subjected to nonlinear constrains in highly dimensional space and can be also applied to solve huge number of different problems. The reason for choosing leopard seals to derive the proposed optimization algorithm is that they are characterized by a very wide search scope whereby each seal roams the ocean individually searching for potential preys which in turn allows the herd to covers a very wide search area [16,20]. However, in the case of one of the leopard seals discovering a potential prey, immediate communication is made between the members of the nearby herd to start chasing and trapping the prey. Hence, they follow the principle that: "Search alone, but hunt together".

LSO is applied in three sequential phases, which are; (i) searching, (ii) encircling, and (iii) attacking. During the searching phase, search agents try to discover the search space (e.g., perform exploration). LSO has a unique ability to detect the search domain, where the search agents move in random spiral movements to cover most of the search range. Of course, this produces many good solutions in preparation for the implementation of exploitation in the following phases, which are encircling and attacking. The number of new solutions produced through every iteration of the search phase can be tuned in advance providing a complete control of the algorithm exploration. LSO provides also a good exploitation through the successive iterations of the encircling and attacking phases. Hence, Leopard Seal Optimization (LSO) is provided to solve many different real time optimization problems and give quasi optimal solutions.

The advantages of the proposed LSO algorithm upon other meta-heuristics are as follows:

(i) The design of LSO depends on the simulation of the hunting strategy of the leopard seals.
(ii) LSO is easy to implement because of its simple natural algorithmic structure.
(iii) The leopard seal's tremendous search ability can exponentially enrich the exploration of the proposed optimization algorithm, which could have the greatest effect in solving the local optima problem that most bio-inspired optimization algorithms suffer from.
(iv) The structure of LSO does not need any particular changes, thus, it is flexible in implementing on different optimization problems.
(v) LSO also has the advantages of not needing gradient information, not depending on the problem model, can obtain a good balance between computational cost and the quality of solution, and it has powerful search ability and is widely applicable.
(vi) LSO is a fast convergence algorithm that does not require any parameter controlling, can provide good global search because it has a good balance between exploitation and exploration.
(vii) LSO can be utilized to maximize or minimize several fitness functions.
(viii) LSO can be applied in optimization applications such as clustering, engineering science, data mining, and medical science.

LSO has been compared against recent six well-known optimization algorithms considering feature selection as the optimization problem. Also, the results of the LSO are compared with the results of these six algorithms. In the results section, the comparative analysis proved that LSO performs very well using four metrics called Wilcoxon test (p-value), t-test, accuracy, and execution time. Hence, a set of experiments based on 5 benchmark functions have been performed to statistically analyze the proposed LSO against other recent algorithms using p-value and t-test metrics. Then, a set of experiments for determining the most informative features to accurately diagnose Covid-19 cases have been applied using accuracy, and execution time metrics. Experimental results proved that LSO outperforms the other competitors based on p-value and t-test and also it can determine the best features for the diagnosing process. This can be concluded from the statistical analysis in terms of p-value and t-test and also from diagnosing result in terms of accuracy and execution time.

In this paper, the structure is organized as follows; the meta-heuristic natural inspired optimization algorithms are provides in Section 2. The problem definition as well as the suggested solution is introduced in Section 3. On the other hand, related work is provided in Section 4. Section 5 discusses the characteristics of leopard seal. The proposed leopard seal optimization algorithm is provided in Section 5. In Section 6, feature selection using leopard seal optimization as a case study is discussed. Discussion of results has been provided in Section 7. At the end, conclusions and future research are provided in Section 8.

2. Meta-heuristic natural inspired algorithms

Lately, the increase in the complexity of real-life problems has increased the need for better meta-heuristic natural inspired algorithms, which are used to obtain the optimal possible solutions to realistic engineering problems during a course of iterations. These algorithms are becoming more common due to their outstanding performance compared to existing classical optimization techniques. As illustrated in Fig. 1, NIO can be categorized into nine main types depending on the inspiration source. These types are Biology Based Algorithms (BBA), Physics Based Algorithms (PBA), Chemistry Based Algorithms (CBA), Social Based Algorithms (SBA), Swarm Intelligence Algorithms (SIA), Music Based Algorithms (MBA), Sports Algorithms (SA), Mathematics Algorithms (MA), and Hybrid Algorithms (HA) [11–14]. In BBA, initial solutions are randomly generated and updated across subsequent generations using crossover and mutation. As the best suited solutions of the current generation have a greater chance of co-producing a new solution in the next generation, BBA methods use the best solution from the previous generation to create new solutions in the next generation [5,21]. Accordingly, over the series of iterations, the initial random solution is optimized. The most popular BBA is Genetic Algorithm (GA) [21]. However, there are several used BBA algorithms such as; Evolutionary Programming (EP) [22], Genetic Programming (GP) [21], Evolution Strategy (ES) [22], and Differential Evolution (DE) [21].

PBA and CBA methods optimize the initial solutions over the series of iterations by mimicking the laws of physics and chemistry as the basic sources of inspiration. Such laws include gravity, river systems, electrical charges, weights, gravitational force, inertia force, and ray casting etc. Some popular PBA are Big Bang Big Crunch (BBBC) [23], Gravitational Local Search (GLS) [24,25], Water Wave Optimization (WWO) [15], and Black Hole Optimization (BHO) [24–26]. Some popular CBA are Atomic Orbital Search (AOS) algorithm [11,12] and Chemical Reaction Optimization (CRO) algorithm [27]. SIA relates to the presence of many interacting agents who follow specific rules that contribute greatly to the success and survival of the herd. SIA algorithms are inspired from the social intelligence and collective behavior of herds, flocks, or swarms. Although each individual agent may not be considered intelligent, the herd as a whole behaves intelligently. The reason for this is the self-organization behavior of the herd members, which contributes greatly as the source of the so-called collective intelligence. Among SIA methods, Particle Swarm Optimization (PSO) is the most popular one [5,15]. However, there are many other SIA methods that have proven successful in solving many real world optimization problems, such as; Ant Colony Optimization (ACO) [15], Bee Colony Optimization (BCO) [5], Moth Flame Optimization (MFO) [15], Grey Wolf Optimization (GWO) [10], Bat Algorithm (BA) [15], and others [5,21].

MBA methods optimize the initial solutions over the series of iterations by mimicking music inspired search. Some popular MBA are Harmony Search Algorithm (HSA) [28] and Stochastic Paint Optimizer (SPO) algorithm [11,12]. SA methods simulate processes, rules, concepts, and events in several sports. Some popular SA are Tiki-Taka Algorithm (TTA) [17] and Most Valuable Player (MVP) algorithm [29]. MA methods mimic the mathematical processes and operations. Some popular MA are Chaos Game Optimization (CGO) algorithm [11,12] and Arithmetic Optimization Algorithm (AOA) [30]. Finally, HA represents algorithms that combines two or more than two types from these nine types of algorithms. Colonial Competitive Differential Evolution (CCDE) algorithm is HA that can be categorized as both BBA and SBA [11].

Recently, SIA algorithms are the most popular and widely used NIO algorithms because; (i) SIA algorithms are efficient as they rely on the interaction and accordingly information sharing among multiple agents [5,21,31]. This allows; co-evolution, self-organization, and learning during iteration, (ii) from the implementation point of view, SIA algorithms are suitable for large scale optimization problems since several agents can be easily parallelized, (iii) SIA algorithms are both stochastic and based on populations. Hence, they introduce better solution quality compared with conventional optimization methodologies, (vi) SIA algorithms have simple structure and require limited parameters for their functioning, (v) moreover, no initial guess is required by SIA algorithms for solving any optimization problem. In spite of the extensive studies and developments, several critical issues concerning SIA algorithms are still opened such as; (i) basically, parameters of most SIA algorithm are set by experience, (ii) SIA optimization algorithms are not supported by an exact and sophisticated theories, they still lack a unified mathematical infrastructure which permits to exactly analyze

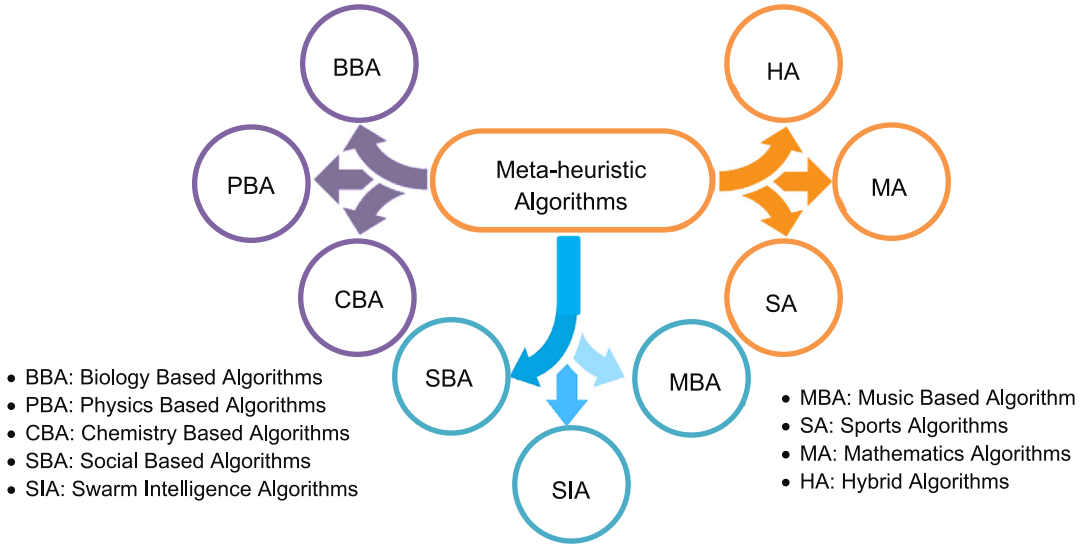


Fig. 1. The nine classes types of meta-heuristic algorithms.

Table 1
The five fundamental principles of swarm intelligence.

Principle	Description
Proximity	The population should be able to carry out simple space and time computations.
Quality	The population should be able to respond to quality factors in the environment.
Diverse response	The population should not commit its activity along excessively narrow channels.
Stability	The population should not change its mode of behavior every time the environment changes.
Adaptability	The population should be able to change its behavior mode when it is worth the computational price.

and compare them, (iii) there is a high reliance of many SIA optimization algorithms on the application environment, (vi) in spite of their effectiveness, the scope of each SIA algorithm is relatively narrow, hence, there is no generally applicable SIA optimization algorithms [5,21,31]. Therefore, introducing appropriate and high scope SIA optimization algorithms becomes imperative. SIA is based on five principles, which are illustrated in Table 1.

As illustrated in Fig. 2, applying a swarm intelligence based algorithm is carried through five steps, which are; (i) a random initialization of a population of individuals in the considered solution space, (ii) defining the stopping criteria (stopping condition), (iii) evaluating the solution in hand through the used fitness function to obtain the current fitness value, (vi) updating the swarm (the considered population of individuals) to obtain new solutions, (v) as the quality of the solution should be improved over iterations. By other words, the fitness value of the solution given time $(t + 1)$ should be better than the fitness value at time (t), hence, after a certain number of iterations, the global best solution can be achieved. In spite of their high efficiency in producing near optimal solutions for real-world optimization problems, no swarm intelligence algorithm is suited for all types of problems. Accordingly, there is still a critical need for more efficient and scalable swarm intelligence optimization algorithms.

Generally, the main distinguishing features of a good NIO algorithm are; (i) higher convergence rate, (ii) unbiased exploration and exploitation, (iii) lower processing time, and (vi) fewer control parameters [15,21]. The convergence rate is the speed in terms of the number of iterations after which the algorithm produces a repeated sequence, which is known as the convergent sequence and is closer to the optimal solution. Exploration and exploitation are the basic parameters of all NIO algorithms. In exploration, new areas of the search space are explored. Exploitation, on the other hand discovers the neighborhood region of previously visited regions of the search space. The processing time is the time required for executing the algorithm. Two types of control parameters are required for any NIO algorithm, namely; (i) Common Control Parameters (CCP), also called regular parameters, and (ii) Algorithm Specific Control Parameters (ASCP), also called dependent parameters. CCPs are the problem independent parameters such as; population size, number of iterations, and number of dimensions. On the other hand, ASCPs are the parameters that are dependent on the problem. Hence, the value of those parameters may vary from problem to another. For illustration, PSO requires learning factors and inertia weight, GA requires mutation and crossover, etc. [21,22]. With no doubt, ASCPs strongly influence the performance of the algorithm. Hence, it is preferred that the NIO algorithm to use as minimum ASCPs as possible.

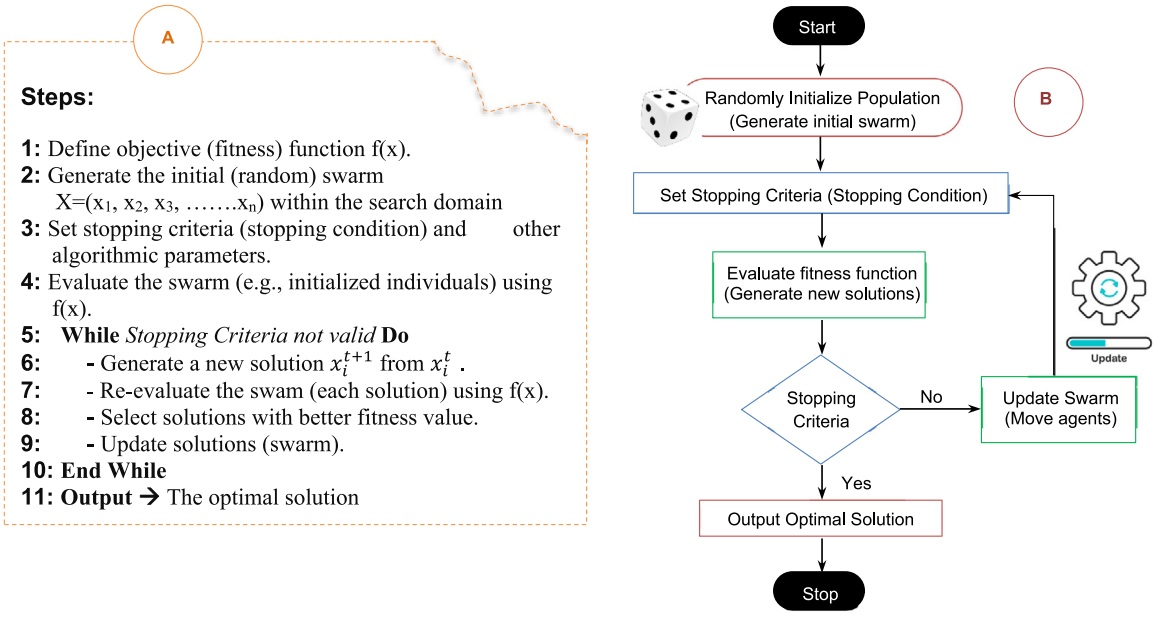


Fig. 2. The basic steps of swarm Intelligence. (A) Pseudo code (B) Flowchart.

3. Problem definition and suggested solution

One might be tempted to ask: "Since there are already have so many natural inspired algorithms, is there is a need to add more?". Although existing NIO algorithms show exceptional power in solving many complex problems, they still suffer from many shortcomings that range from premature convergence, search space inefficient exploration, use of too many parameters, and a complex fitness function, which delays getting the desired solution. Therefore, Leopard Seal Optimization (LSO) is an attempt to complement existing algorithms. LSO aims to solve the weaknesses of earlier NIO algorithms, especially delay and inefficiency problems. We have chosen leopard seal to suggest an optimization method that mimics their social behavior. Leopard seals have excellent ability to communicate with each other in an intelligent manner with a high level of cooperation. They have unique and distinct tactics, whether in searching, chasing, or attacking the prey, which makes them a rich subject for deducing a strong optimization technique that can be used to solve complex problems.

4. Related work

Nature-inspired algorithms are frequently employed to find solutions for several optimization problems. Several researchers have suggested a huge number of nature-inspired algorithms during the last few decades. Some of these algorithms have proven to be quite efficient when compared to other traditional optimization approaches. Not every algorithm is suitable for every type of problem. Some algorithms perform better than others. In this section, an attempt has been made to outline various SIA methods that belong to the same class category as the proposed LSO algorithm.

In [32], Tuna swarm optimization (TSO) as a new SIA algorithm was introduced to mimic the cooperative foraging behavior of tuna swarms by simulating two natural foraging behaviors called spiral foraging and parabolic foraging. In fact, TSO algorithms are fewer parameters, fast compared to other recent algorithms, and simple structure. On the other hand, it may be further accelerated and also may fall into local optima when it is used to solve complex problems. As presented in [33], Pelican Optimization Algorithm (POA) that simulates the behavior and strategy of pelicans when assaulting and hunting prey was provided as new SIA algorithm. In fact, POA can solve real-time engineering problems such as speed reducer design, pressure vessel design, etc. However, POA needs to be improved for increasing its accuracy and convergence speed and also for providing global optimization solution without falling into local optima.

In [34], Cat and Mouse-Based Optimization (CMBO) that is derived from the natural behaviors of cats hunting a mouse and a mouse escaping to shelters was presented as a new SIA algorithm. Actually, CMBO can provide global optima solutions but it cannot provide fast solutions. Additionally, it provide a high cost to solve problems. Related to [35], Aphid–Ant Mutualism (AAM) as a new SIA algorithm that inspired by a special relation between aphids and ant species was provided to solve the problems of real-time applications. AAM avoids premature convergence and can reach to global optima solutions. Although AAM is a fast algorithm that can provide accurate solutions, its complexity and cost is high and also it cannot be applied in multi-task optimization and multi-objective problems.

Table 2
A comparison between the proposed LSO and many recent optimization algorithms.

Algorithm	Advantages	Disadvantages
Tuna swarm optimization (TSO) [32]	- Fewer parameters. - Fast. - Simple structure.	- Further accelerated. - Falling into local optima.
Pelican Optimization Algorithm (POA) [33]	- Solving real-time engineering problems. - low complexity.	- Low accuracy. - High convergence speed
Cat and Mouse-Based Optimization (CMBO) [34]	- Solving real-time problems. - Providing global optimal solutions	- Low speed. - High cost.
Aphid-Ant Mutualism (AAM) [35]	- Fast. - Providing global optimal solutions.	- High complexity. - High cost. - Cannot be applied in multi-task optimization and multi-objective problems
White Shark Optimizer (WSO) [36]	- Fewer parameters. - High flexibility.	- Cannot reach to global optimal solution
Flamingo Search Algorithm (FSA) [37]	- Simple. - low complexity.	- Falling into local optima. - Low accuracy.
Red Piranha Optimization (RPO) [38]	- Simple. - Accurate and fast . - Fewer parameters - Solving real-time engineering problems.	- Has difficulty locating prey.
Leopard Seal Optimization (LSO)	- Flexible. - Simple. - Fewer parameters. - Accurate. - Fast. - Solving real-time engineering problems.	- Needs to further improve performance by hybridizing the LSO with other algorithms. - Needs to be applied on many different applications.

As mentioned in [36], White Shark Optimizer (WSO) influenced by great white shark activities, particularly by how they navigate and forage using their superior hearing and smell senses was provided as a new SIA algorithm. In fact, WSO are fewer parameters and has a high flexibility to find solutions for several different optimization problems. While WSO is fast and can provide accurate solutions, it cannot reach to global optimal solution. In [37], a new SIA algorithm called Flamingo Search Algorithm (FSA) that is derived from the natural behavior of flamingos that is exemplified in migrating and foraging. FSA is a simple algorithm that can prove a new solutions for different problems. On the other hand, it may fall into local optima and cannot provide accurate solutions.

According to [38], Red Piranha Optimization (RPO) as a new SIA algorithm that mimics the cooperation behavior of the red piranha fish for hunting or saving their eggs was provided to solve complex engineering problems. RPO can be applied as a binary version to determine the best features before using the diagnosis model to introduce accurate results. RPO is a simple algorithm that can provide accurate and fast solutions but it has difficulty locating prey. During this paper, we introduced a new SIA algorithm called Leopard Seal Optimization (LSO) that simulates the hunting strategy of the leopard seals to reach to global optimal solutions for complex engineering problems that may be difficult for the most bio-inspired optimization algorithms. Actually, LSO is a flexible algorithm that have wide applicability and strong search capability, and can obtain a good balance between computational cost and the quality of solution. LSO can provide fast and accurate solutions without requiring any parameter controlling. Although LSO can overcome several problems of other recent SIA algorithms, it should be tested on many different real-time case studies or applications. Also, hybridization of LSO with other algorithms may be considered to further improve performance. A comparison between the proposed LSO and these recent SIA optimization algorithms are presented in Table 2.

5. Leopard seals as intelligent and co-operative animals

Leopard seal is a massive and fearsome aquatic dominant predator of Antarctica. Over millions of years of evolution, they have acquired specific adaptations to cope with the harsh realities of its cold marine environment. Leopard seal can grow more than 1200 pounds and longer than 11 ft. Although leopard seals are very clumsy creatures on the land, they have fast swimming speed. Moreover, with their sharp teeth and strong jaws, the leopard seal can do serious damage to their prey. As generalist apex predators, leopard seals able to eat almost anything. However, recent studies have shown that their diet mainly consists of krill, small seals, and penguins [16].

Fig. 3. Leopard seals hunting and chasing their prey.

Leopard seals are smart, wonderful animals that spend most of their lives searching for food in the ocean. The strange thing is that these amazing animals like to search for prey, as they may chase the prey until it dies or escapes, and even after killing the prey, they may not devour it. That behavior that puzzled scientists and the only explanation is that the leopard seals are animals that like to search for prey and chase them, even if for the sake of sport. Another distinct behavior of leopard seals is that while searching for food, they distribute themselves so that each individual is responsible for the search within a specific range. This behavior maximizes the search coverage compared with the search within a group directed by a single leader. When an individual discovers a potential prey, it summons the rest of the herd members present in the vicinity to start chasing the prey. Therefore, leopard seals search individually but attack together [16,20]. Fig. 3 shows how leopard seals hunting and chasing their prey.

6. The proposed Leopard Seal Optimization (LSO)

Leopard Seal Optimization (LSO) is a NIO algorithm that simulates the hunting behavior of the leopard seals. Leopard seals are smart animals that cooperate for the benefit of all. The exceptional thing is that the leopard seals like to search for food and chase prey, even if it is not hungry, so often after killing the prey, it does not devour it. There are three activities (phases) associated with leopard seal hunting, which are: searching, encircling, and attacking as shown in Fig. 4. During the search phase, leopard seals search individually for food to cover as wide a search area as possible. Each individual searches independently to cover a region of his own, so that it takes its decision in the next movement by himself, and no one leads him or determines the direction of his next movement. This allows a small herd to cover a very large search area. Once an individual finds a potential prey, it issues a specific signal called "Encircle message" in all directions to summon the herd members present in the vicinity, informing them of the presence of potential prey. Through the strength and direction of the signal received by individuals in the vicinity, the neighboring individuals can determine the location of the individual who sent the signal, then, they head immediately to the specified place to start chasing the prey and encircle it. After completing the surrounding of the prey, the prey becomes unable to maneuver or move, as the attack phase begins. The three phases are mathematically modeled in the following subsections.

6.1. Searching for a prey

Generally, exploitation is to make the best decision given the information in hand, while, exploration is to collect more information. Hence, it will be better to collect enough information to make the best overall decisions. Exploration is done mainly during the search phase, while encircling and attacking are dedicated for exploitation. Beyond any doubt,

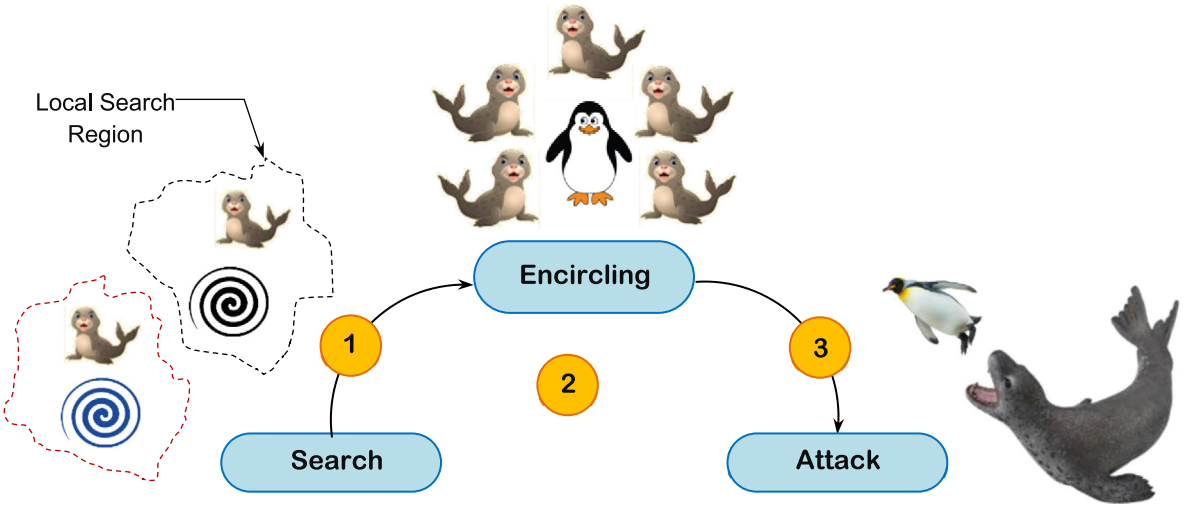


Fig. 4. LSO general sequential phases.

the success of the search phase at which the algorithm performs exploration, leads to the success of all subsequent steps of the algorithm, at which the algorithm seeks to achieve exploitation. Conversely, if the search phase fails, the algorithm will fail into a Local Optima Problem (LOP). A Local Optima (LO) is the extrema of the objective function (minimum or maximum) for a given region of the input search space, while the optimal solution, which is the target of the algorithm is considered as the Global Optima (GO). The main cause of the LOP is that the search agents cannot extensively detect most parts of the search domain during the search phase as illustrated in Fig. 5. LOP will have a significant effect on the algorithm's performance if those unexplored parts of the search space contain the target global optima. As illustrated in Fig. 5, during the exploitation stage an attraction force will occur between the different search agents on one hand and the local optima points on the other hand. Hence, the algorithm will ignore the optimal solution that it actually searches for. Therefore, the algorithm will give a set of potential solutions that may be far from the optimal one.

In all natural inspired optimization algorithms, searching phase is dedicated for exploration. To accomplish such aim, the search agents are allowed to move randomly to discover the search space as much as possible. Although the movements of the search agents are considered as random, the agents' movements are guided by random leader(s). In the proposed LSO, a self-guidance procedure is used to guide the movement of each search agent. The employed movement methodology of the search agents during the search phase allows a perfect coverage of the search domain, which in turn guarantees a successful exploration.

Although, the search for prey is often carried out individually, many successful cooperative hunting cases have been observed. What is distinctive about this strategy is that leopard seals choose a solitary search for prey so that they can scan the widest area and thus discover many potential preys. If we imagine a herd of ten individuals, if the herd moves as one mass in a specific direction, then the chance of finding potential prey will certainly be much less (it may reach one over ten) than in the case of each agent searching individually. Leopard seals devised a different strategy for hunting prey. Fig. 6, gives an illustration considering 6 search agents. In the upper of Fig. 6, it can be concluded that when the group moves randomly in one consistent block, it will not able to scan the whole search space efficiently. On the other hand, as illustrated in the lower part of Fig. 6, when each agent moves individually in a random way, the search space will be better scanned. This guarantees that the optimal solution can be reached by the algorithm.

The effective leopard seal hunting strategy is based on the principle "search individually, but hunt as a group". According to this strategy, each individual searches with a self-guided strategy, and thus search agents can cover the largest area of the search domain during the search phase. When an agent finds a prey, he signals to his peers that he has found a potential prey. Thus, most of the potential prey can be identified within the scope of the research domain, and after the search is over, a dialogue between the members of the group begins to encircle the best prey discovered.

In addition to search individually and randomly, each agent tries to cover as much space as it roams. The discovered movement pattern of a roaming leopard seal during the search for a prey can be described as a random movement with rotation. It moves randomly and rotates simultaneously as illustrated in Fig. 7. The movement of the search agent through one iteration is similar to the spoon in shape, which consists of two main parts, namely; (i) Spoon Arm (SA) and (ii) Spoon Head (SH).

The position vectors of the leopard seal during its movement in each iteration, which are ξ position vectors, are recorded. The start point of the spoon arm of the ith iteration, denoted $\vec{X}^{i}_{init}$, is known in advance as it represents the

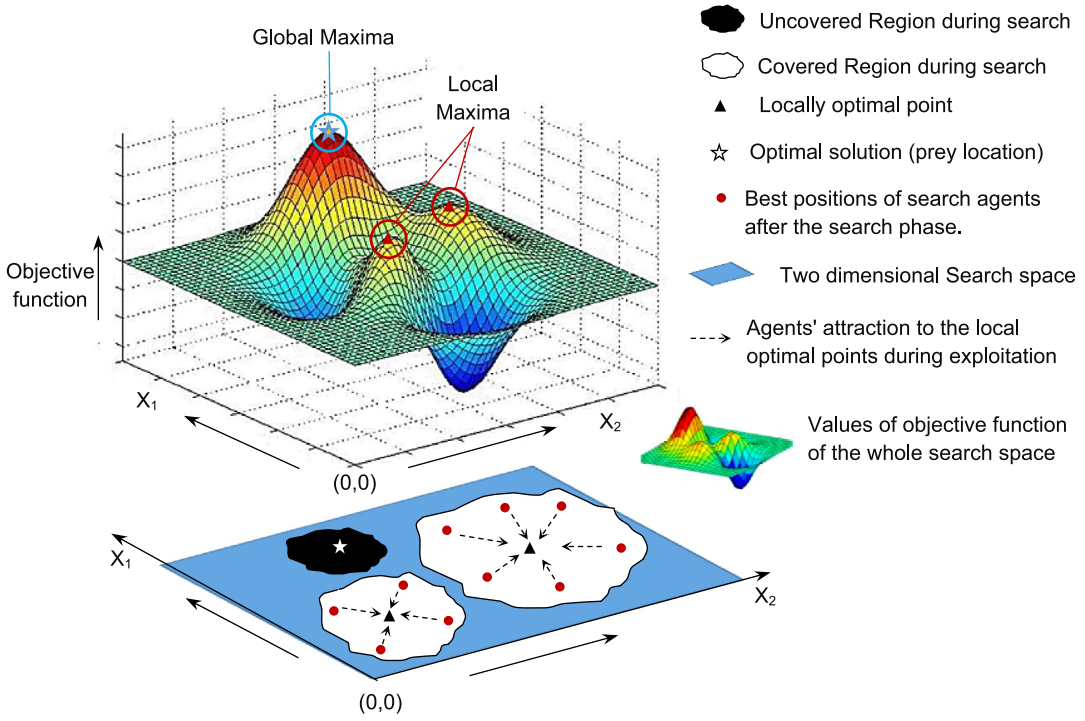


Fig. 5. The local optima problem and uncovered regions of the search space.

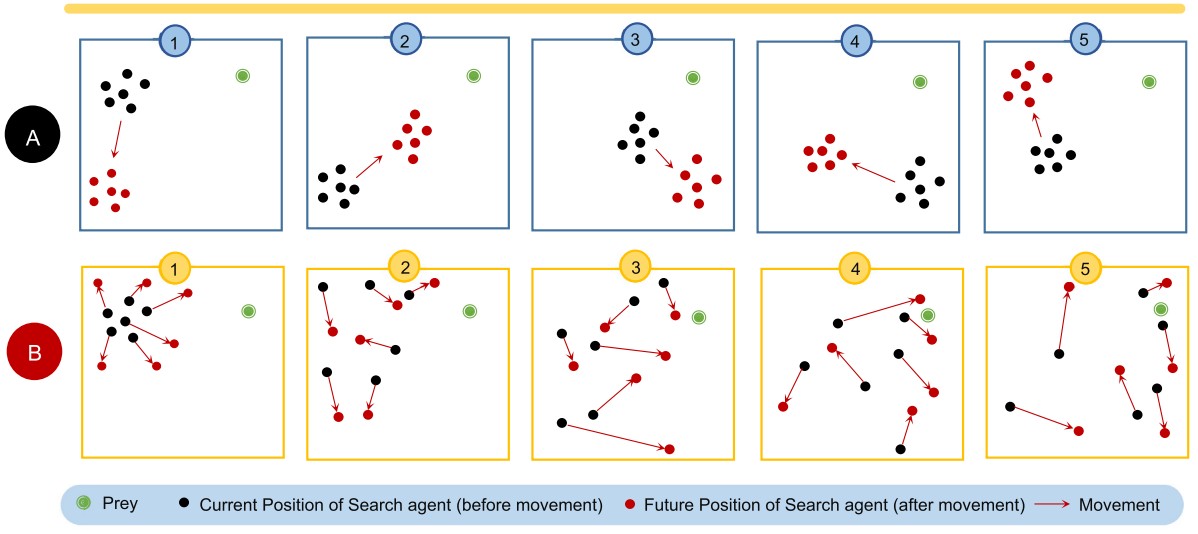


Fig. 6. Group versus individual movements of the search agents (A) Group movement (B) individual movements.

end point of the spiral of the (i − 1)th iteration, so it can be also denoted as; $\vec{X}_{\xi}^{i-1}$ (note, $\vec{X}_{init}^{i}$ can be also a random position vector in the case of the first iteration, e.g., if i = 1). On the other hand, the end point of spoon arm of the *i*th iteration, denoted $\vec{X}_{1}^{i}$ is set randomly and represents the start of the spiral of the current iteration. The end point of the

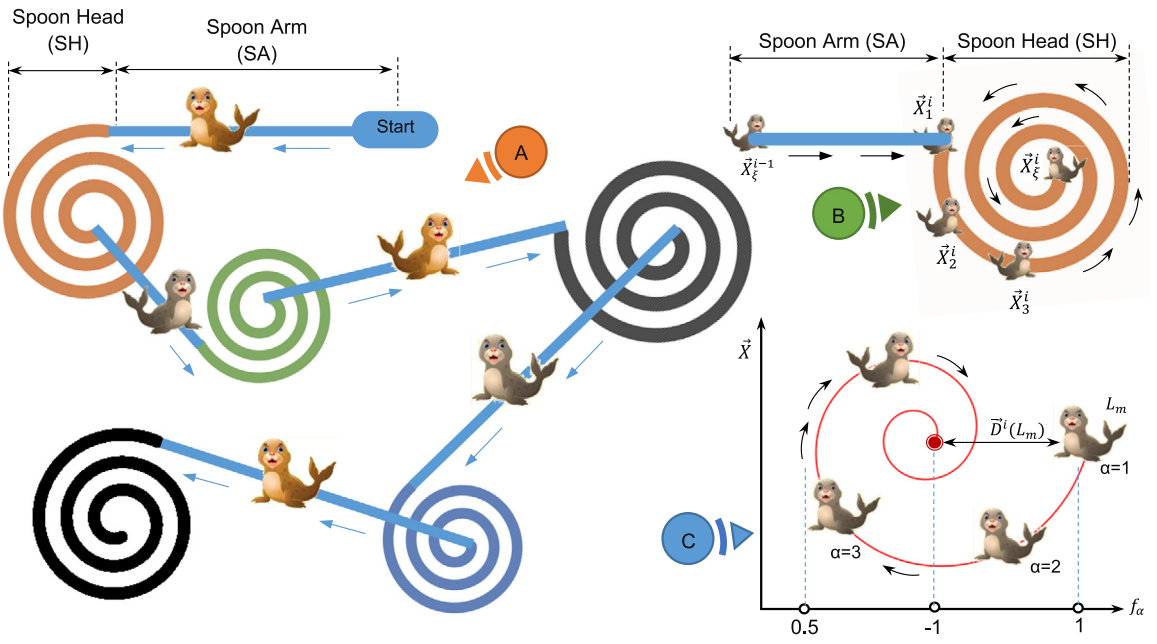


Fig. 7. The movement pattern of the leopard seal during the search phase. (A) Movement through five iterations (B) The movement during one *i*th iteration (C) Movement within the spoon head.

spiral of the *i*th iteration is denoted as; $\vec{X}_\xi^i$, which is set also randomly. Hence, during each iteration of the search phase, we have ξ position vectors for the roaming leopard seal, denoted as; $\vec{X}_p^i \ \forall \ p \in \{1,2,3,\ldots, \xi\}$, as well as the corresponding objective function value $F(\vec{X}_p^i) \ \forall \ p \in \{1,2,3,\ldots, \xi\}$. It is important to mention that ξ is an arbitrary number that sets the degree of smoothness of SH, hence, the more the value of ξ, the more SH smoothness. If the value of ξ is small, this minimizes the covered area by the leopard seal during the search phase. On the other hand, if the value of ξ is high, this increases time delay to finish the search phase of the algorithm. The ξ position vectors of the *i*th iteration for the leopard seal L_m can be calculated using (1) [39].

$$\vec{X}_\alpha^i\,(L_m)\,|_{\forall\,\alpha\in 2,3,4,\ldots,\xi-1} = \vec{D}^i\,(L_m)\,.e^{bf_\alpha}\cos(2\pi f_\alpha) + \vec{X}_\xi^i\,(L_m) \tag{1}$$

where $\vec{X}_\alpha^i\,(L_m)$ is the updated α's position of L_m at *i*th iteration and α indicates to positions of L_m from 2 to $\xi - 1$; $\alpha \in \{2, 3, 4,\ldots, \xi - 1\}$. The distance between the leopard seal and the prey is $\vec{D}^i\,(L_m)$ that can be calculated using (2) while the logarithmic spiral shape is $b.f_\alpha$ is the angle scaling factor for the α's position of the leopard seal that can be calculated using (3) and $\vec{X}_\xi^i\,(L_m)$ is the updated ξ's position (the last position) of L_m at *i*th iteration.

$$\vec{D}^i(L_m) = \left|\vec{X}_\xi^i(L_m) - \vec{X}_1^i(L_m)\right| \tag{2}$$

$$f_\alpha = 1 - \frac{2 * \alpha}{\xi}(3) \tag{3}$$

where $\vec{X}_1^i(L_m)$ is the updated 1st position of L_m at *i*th iteration, α indicates to the current position of L_m, and ξ is the total positions number of L_m. As all natural inspired optimization algorithms, LSO tries to determine the optimal solution through a set of consecutive iterations. The total number of allowed iterations, denoted as; *N*, is distributed among searching, encircling, and attacking phases as provided in (4).

$$N = N_{Srch} + N_{Enc} + N_{Att} \tag{4}$$

where N_{Srch}, N_{Enc}, and N_{Att} are the iterations number for searching, encircling, and attacking which can be measured applying (5)–(7).

$$N_{Srch} = \left\lfloor \frac{N}{3} \right\rfloor \tag{5}$$

$$N_{Enc} = \left\lfloor \frac{N}{3} \right\rfloor \tag{6}$$

Table 3
The three iterations of the search phase.

	Initially		First Iteration										
	$\vec{X}^{\square}_{init}(L_m)$	$F(\vec{X}^{\square}_{init}(L_m))$	$\vec{X}^1_1(L_m)$	$\vec{X}^1_5(L_m)$	$\vec{D}^1_{\square}(L_m)$	$\vec{X}^1_2(L_m)$	$\vec{X}^1_3(L_m)$	$\vec{X}^1_4(L_m)$	Objective Function				
	$\binom{x_1}{x_2}$		$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$		$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$	$F(\vec{X}^1_1(L_m))$	$F(\vec{X}^1_2(L_m))$	$F(\vec{X}^1_3(L_m))$	$F(\vec{X}^1_4(L_m))$	$F(\vec{X}^1_5(L_m))$
L1	$\binom{2.693}{1.233}$	18.770	$\binom{0.093}{1.996}$	$\binom{2.922}{0.761}$	$\binom{2.829}{1.235}$	$\binom{-0.416}{-0.696}$	$\binom{0.679}{-0.218}$	$\binom{1.803}{0.272}$	14.977	-0.248	4.140	10.528	18.782
L2	$\binom{-1.782}{4.721}$	52.196	$\binom{1.223}{3.031}$	$\binom{-0.6312}{2.981}$	$\binom{1.854}{0.050}$	$\binom{-2.425}{2.922}$	$\binom{-2.102}{2.941}$	$\binom{-1.365}{2.961}$	24.546	31.341	29.812	26.788	24.828
L3	$\binom{4.233}{-0.193}$	29.467	$\binom{3.811}{-0.931}$	$\binom{1.331}{4.981}$	$\binom{2.480}{5.912}$	$\binom{-1.068}{-1.994}$	$\binom{-0.636}{0.293}$	$\binom{0.350}{2.642}$	25.837	-4.125	3.576	20.446	45.538
L4	$\binom{3.117}{-2.677}$	23.752	$\binom{-4.981}{1.933}$	$\binom{-2.332}{1.889}$	$\binom{2.649}{0.044}$	$\binom{-4.894}{1.837}$	$\binom{-4.433}{1.854}$	$\binom{-3.380}{1.872}$	38.945	36.880	32.856	24.980	19.304
L5	$\binom{-0.988}{3.873}$	36.319	$\binom{-1.568}{0.023}$	$\binom{0.842}{2.031}$	$\binom{2.410}{2.008}$	$\binom{-1.489}{-0.338}$	$\binom{-1.069}{0.439}$	$\binom{-0.112}{1.237}$	2.451	0.498	4.421	9.403	15.932
L6	$\binom{-3.644}{-0.274}$	6.971	$\binom{2.091}{-3.123}$	$\binom{-0.421}{1.771}$	$\binom{2.512}{4.894}$	$\binom{-2.851}{-4.003}$	$\binom{-2.413}{-2.110}$	$\binom{-1.415}{-0.165}$	15.346	-5.974	-5.083	1.303	13.301

	Second Iteration										
	$\vec{X}^2_1(L_m)$	$\vec{X}^2_5(L_m)$	$\vec{D}^2_{\square}(L_m)$	$\vec{X}^2_2(L_m)$	$\vec{X}^2_3(L_m)$	$\vec{X}^2_4(L_m)$	Objective Function				
	$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$		$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$	$F(\vec{X}^2_1(L_m))$	$F(\vec{X}^2_2(L_m))$	$F(\vec{X}^2_3(L_m))$	$F(\vec{X}^2_4(L_m))$	$F(\vec{X}^2_5(L_m))$
L1	$\binom{-3.234}{1.112}$	$\binom{-1.099}{-0.066}$	$\binom{2.135}{1.178}$	$\binom{-3.618}{-1.455}$	$\binom{-2.792}{-0.999}$	$\binom{-1.944}{-0.531}$	16.272	-0.112	-0.577	0.015	1.680
L2	$\binom{-0.612}{-0.345}$	$\binom{1.809}{0.877}$	$\binom{2.421}{1.222}$	$\binom{-0.533}{-0.565}$	$\binom{-0.111}{-0.092}$	$\binom{0.851}{0.394}$	0.678	-0.022	2.421	6.821	12.581
L3	$\binom{0.005}{0.098}$	$\binom{-0.087}{4.987}$	$\binom{0.092}{4.889}$	$\binom{-0.176}{-0.781}$	$\binom{-0.160}{1.110}$	$\binom{-0.123}{3.053}$	3.412	0.028	8.556	24.676	48.086
L4	$\binom{-1.928}{1.053}$	$\binom{3.741}{-1.223}$	$\binom{5.669}{2.276}$	$\binom{-1.742}{-3.908}$	$\binom{-0.754}{-3.028}$	$\binom{1.498}{-2.124}$	10.209	-4.618	-3.168	7.437	25.656
L5	$\binom{0.073}{3.091}$	$\binom{-2.099}{-0.988}$	$\binom{2.172}{4.079}$	$\binom{-4.20}{-5.000}$	$\binom{-3.821}{-4.223}$	$\binom{-2.958}{-2.602}$	24.844	-3.760	-5.236	-5.500	-1.842
L6	$\binom{-4.008}{-3.711}$	$\binom{-4.112}{2.309}$	$\binom{0.104}{6.02}$	$\binom{-4.213}{-4.793}$	$\binom{-4.194}{-2.465}$	$\binom{-4.153}{-0.073}$	-4.898	-4.069	-1.918	11.355	35.747

	Third Iteration										
	$\vec{X}^3_1(L_m)$	$\vec{X}^3_5(L_m)$	$\vec{D}^3_{\square}(L_m)$	$\vec{X}^3_2(L_m)$	$\vec{X}^3_3(L_m)$	$\vec{X}^3_4(L_m)$	Objective Function				
	$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$		$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$	$\binom{x_1}{x_2}$	$F(\vec{X}^3_1(L_m))$	$F(\vec{X}^3_2(L_m))$	$F(\vec{X}^3_3(L_m))$	$F(\vec{X}^3_4(L_m))$	$F(\vec{X}^3_5(L_m))$
L1	$\binom{-0.811}{0.003}$	$\binom{-0.772}{2.833}$	$\binom{0.039}{2.83}$	$\binom{-0.818}{-0.506}$	$\binom{-0.803}{0.589}$	$\binom{-0.787}{1.713}$	2.050	-0.148	5.214	13.183	23.597
L2	$\binom{-3.866}{2.914}$	$\binom{3.871}{0.092}$	$\binom{7.737}{2.822}$	$\binom{-3.613}{-3.237}$	$\binom{-2.264}{-2.146}$	$\binom{0.810}{-1.024}$	41.627	-5.338	-5.239	3.058	25.747
L3	$\binom{0.054}{-3.456}$	$\binom{1.113}{-2.087}$	$\binom{1.059}{1.369}$	$\binom{0.089}{-3.702}$	$\binom{0.273}{-3.173}$	$\binom{0.694}{-2.629}$	1.418	2.411	1.863	3.089	4.795
L4	$\binom{1.231}{-2.551}$	$\binom{-2.913}{-4.544}$	$\binom{4.144}{1.993}$	$\binom{-5.000}{-5.000}$	$\binom{-5.000}{-5.000}$	$\binom{-4.553}{-5.000}$	6.421	-2.000	-2.000	-3.142	-5.105
L5	$\binom{-1.223}{0.801}$	$\binom{3.188}{-0.088}$	$\binom{4.411}{0.889}$	$\binom{-1.079}{-1.137}$	$\binom{-0.310}{-0.793}$	$\binom{1.443}{-0.440}$	6.875	-2.475	-0.312	7.036	19.476
L6	$\binom{-0.081}{0.923}$	$\binom{2.443}{1.119}$	$\binom{2.524}{0.196}$	$\binom{0.002}{0.888}$	$\binom{0.441}{0.964}$	$\binom{1.444}{1.041}$	7.463	7.341	8.435	11.721	16.849

$$N_{Att} = \left(N - 2 * \left\lfloor \frac{N}{3} \right\rfloor\right) \tag{7}$$

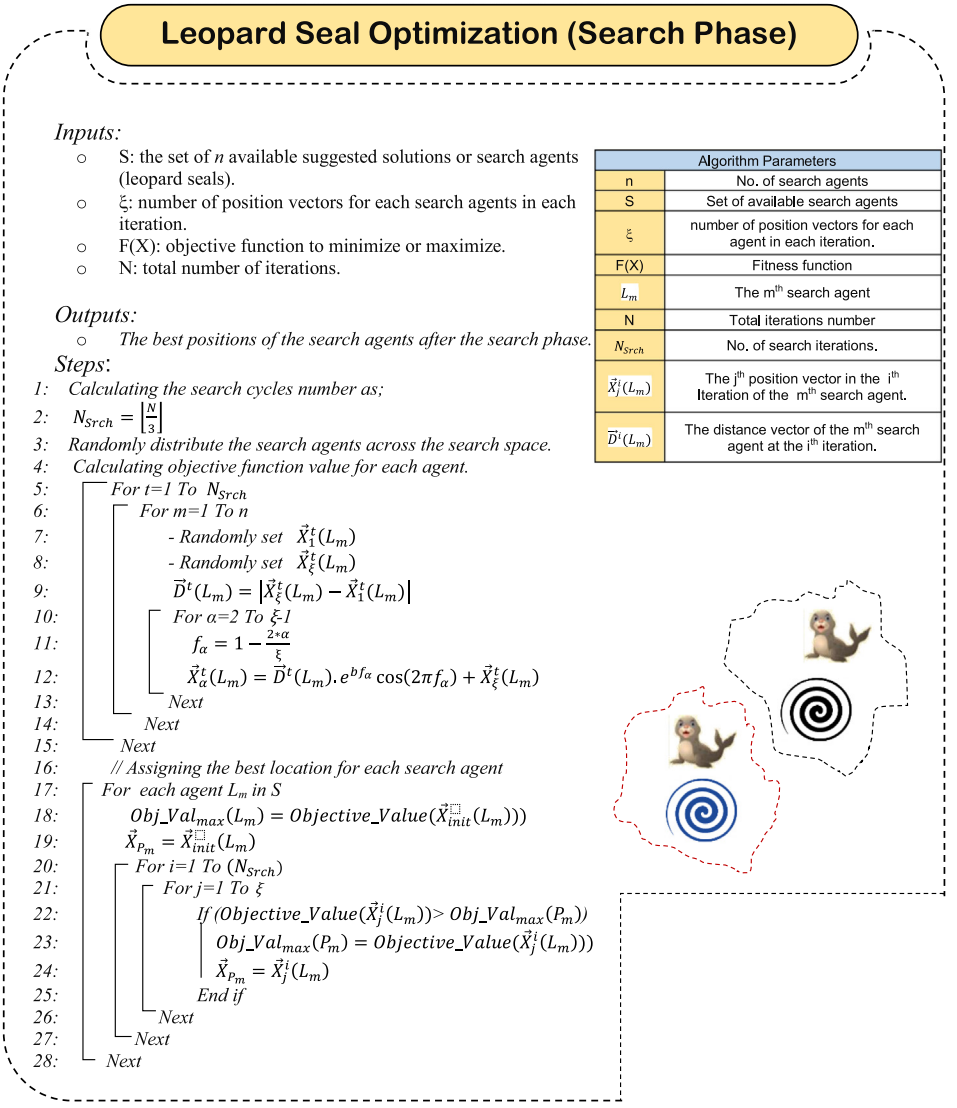
As an illustrative example, it is supposed that there are $n = 6$ search agents (leopard seal) in 2-dimensional space x_1 and x_2. It is supposed that x_1 and $x_2 \in [-5,5]$, which the employed search domain. Supposing the iterations number is 10 (N = 10). Accordingly, $N_{\text{Srch}} = \left\lfloor \frac{10}{3} \right\rfloor = 3$, $N_{\text{Enc}} = \left\lfloor \frac{10}{3} \right\rfloor = 3$, $N_{\text{Att}} = 10 - 3 - 3 = 4$. Also, $\xi = 5$, hence, during each iteration of the search phase, for each search agent L_m there are five position vectors, which are; $\vec{X}^i_{\alpha}(L_m) \mid_{\forall\, \alpha \in 1,2,3,4,5}$, where i refers to the iteration number. The initial vectors of positions for the 6 search agents are randomly determined using (8).

$$x = low + rand * (high - low) \tag{8}$$

where the upper value in the considered interval (e.g., 5) is *high* and the lower value (e.g., −5) is *low*. The fitness function is applied to minimize $f(X) = x_1^2 - x_1x_2 + x_2^2 + 2x_1 + 4x_2 + 3$. During the search phase, the three iterations are summarized in Table 3. Algorithm 1 includes the steps of search phase for LSO.

Table 4
The optimal positions for all search agents in illustrative example.

Search agent	Position		Objective function	Rank
	x_1	x_2		
L_1	−2.792	−0.999	−0.577	6
L_2	−3.613	−3.237	−5.338	3
L_3	−1.742	−3.908	−4.125	5
L_4	−2.913	−4.544	−5.105	4
L_5	−2.958	−2.602	−5.500	2
L_6	**−2.851**	**−4.003**	**−5.974**	**1**



For illustration, consider the value of x_2 in the position vector $\vec{X}_2^2(L_5)$ in which the actual calculated value was −5.8. This value is outside the acceptable range (e.g., $x_2 \notin [-5,5]$). Accordingly, in Table 3, this value is replaced by −5 that represents the minimum legal value. Additionally, in the case if the value of x_1 or x_2 is above the maximum value that equals 5, it needs to be replaces by 5. After the 3 cycles of the search phase are finished, each search agent will includes sixteen different position vectors with fifteen vectors generated by the corresponding fitness function in addition to the initial random position. Leopard seals perform greedy selection because they are greedy predators. Hence, each leopard seal back the discovered positions and the fitness function validation level after the searching phase is finished. Then, leopard seal depends on the best position which provides the best fitness value. Table 4 includes the best position vectors for all leopard seals with the corresponding fitness values.

6.2. Encircling the prey

The search phase allows the search agents to move randomly and irregularly to discover most areas of the search space, and thus select the areas that are higher in achieving the objective function, which represents a possible place for the prey and thus possible places for the optimal solution. The main goal of the encircling phase is to allow search agents to surround the prey that indicates to the best solution. Perhaps the main problem in the encircling phase is that the location of the prey, which is the target place to encircle, is not known in advance. Hence, before prey encircling, the prey position must be identified, which is called prey allocation problem. Three different methods can be followed for prey allocation (e.g., predicting the potential position of the prey), which are; (i) Single Leader Prey Allocation (SLPA), (ii) Multiple Leaders Prey Allocation (MLPA), and (iii) Weighted Leaders Prey Allocation (WLPA).

SLPA is the simplest method for prey allocation. Based on SLPA, the predicted position of the potential prey is the position of the leopard seal that maximizes the fitness function validation, which is considered as the single leader of the leopard seal hunting group as depicted in (9).

$$\vec{X}_{prey} = Position\left[\underset{\forall L_m \in S}{argmax\_Validation}\left[F(L_m)\right]\right] \tag{9}$$

where S is the set of available solutions (e.g., search agents) and $F(L_m)$ is the employed fitness function for the leopard seal L_m and $argmax\_Validation\left[F(L_m)\right]$ returns with the search agent which maximizes the validation of the employed objective function. MLPA depends on selecting a set of leaders so that the rest of the group follows them. The selected leaders are those that introduce the maximum validation of the employed objective function. Assuming there are k selected leaders whose position vectors are; $\vec{X}_1$, $\vec{X}_2$, ... $\vec{X}_k$ in z dimensional space in which the position vector of the ith leader can be represented as; $\vec{X}_i = [x_{1i}, x_{2i}, x_{3i}, \ldots, x_{zi}]$. Hence, the predicted position of the prey can be measured by (10). Hence, the predicted location of the prey represents the average location of the determined leaders.

$$\vec{X}_{prey} = \frac{1}{k}\begin{pmatrix} \sum_{i=1}^{k} x_{1i} \\ \sum_{i=1}^{k} x_{2i} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \sum_{i=1}^{k} x_{zi} \end{pmatrix} \tag{10}$$

In spite of its effectiveness, MLPA is not accurate especially if the actual location of the prey is very close to one of the leaders and very far from the other leaders. This leads to an incorrect prediction of the location of the prey. To compensate the defects of MLPA, a new method for prey allocation is introduced to accurately predict the location of the prey, which is WLPA. According to WLPA, each of the selected leaders is assigned a weight based on the extent of its proximity to the prey, which can be measured by the extent to which the objective function is achieved by this search agent. Again, assuming there are k selected leaders whose position vectors are; $\vec{X}_1$, $\vec{X}_2$, ... $\vec{X}_k$ in z dimensional space in which each leader is assigned a weight. The position vector of the ith leader can be represented as; $\vec{X}_i = [x_{1i}, x_{2i}, x_{3i}, \ldots, x_{zi}]$ and it is assigned the weight w_i. Hence, the predicted position of the prey can be measured by (11).

$$\vec{X}_{prey} = \frac{1}{\sum_{m=1}^{k} W(L_m)}\begin{pmatrix} \sum_{m=1}^{k} W(L_m).x_{1m} \\ \sum_{i=1}^{k} W(L_m).x_{2m} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \sum_{i=1}^{k} W(L_m).x_{zm} \end{pmatrix} \tag{11}$$

Perhaps the important issue now is to determine the mechanism for calculating the weight of search agents. In general, the higher degree of validation of the objective function by the search agent L_m the higher the weight of L_m, as this indicates the L_m's proximity to the prey that represents the optimal solution. A suggested methodology to assign a weight to each

Table 5
The objective function for each search agent, and corresponding weight.

Search agent	$\vec{X}(L_m)$	F(L_m)	Objective = maximize				Objective = minimize			
	$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$		F(L_X)	F(L_Y)	Range = |F(L_Y)–F(L_X)|	Weight	F(L_X)	F(L_Y)	Range = |F(L_Y)–F(L_X)|	Weight
L_1	$\begin{pmatrix} 4 \\ 6 \end{pmatrix}$	3.214				0.521				0.479
L_2	$\begin{pmatrix} 2 \\ 13 \end{pmatrix}$	−6.921				0				1
L_3	$\begin{pmatrix} 12 \\ 2 \end{pmatrix}$	12.542	−6.921	12.542	19.463	1	12.542	−6.921	19.463	0
L_4	$\begin{pmatrix} 1 \\ 9 \end{pmatrix}$	−1.331				0.287				0.713
L_5	$\begin{pmatrix} 7 \\ 10 \end{pmatrix}$	8.453				0.790				0.210

search agent is to map the degree of objective function validation of the search agent to a corresponding weight that ranges from '0' to '1'. Hence, the search agent whose weight equals '0' is the lowest in achieving the objective function, while the search agent whose weight equals '1' is the lowest in achieving the objective function. However, two distinct procedures should be followed as there are two different objectives to satisfy the employed objective function, which may be "minimize" or "maximize". Assuming L_x is the search agent with the lowest validation of the employed objective function and L_y is the search agent with the highest validation. Hence, weight $(L_x) = 0$ and weight $(L_y) = 1$. Hence, for calculating the weight of the search agents, initially, the range is calculated using (12), then, the weight of the search agent L_m, denoted as $W(L_m)$ can be calculated using (13).

$$Range = \left|F\left(L_y\right) - F\left(L_x\right)\right| \tag{12}$$

$$W\left(L_m\right) = \frac{|F(L_m) - F(L_x)|}{Range} = \frac{|F(L_m) - F(L_x)|}{\left|F\left(L_y\right) - F\left(L_x\right)\right|} \tag{13}$$

As an illustrative example, consider five search agents denoted as; L_1, L_2, L_3, L_4, and L_5 in two dimensional space (x_1, x_2). The objective function for each search agent as well as the corresponding weight considering the two objectives (e.g., maximize or minimize) are illustrated in Table 5. By inspecting Table 5, it can be concluded that $W(L_m)|_{Obj=max} = 1 - W(L_m)|_{Obj=min}$, where $W(L_m)|_{Obj=max}$ is the weight of the search agent L_m when it is required to maximize the objective function and $W(L_m)|_{Obj=min}$ is the weight of the search agent L_m when it is required to minimize the objective function.

Based on the data illustrated in Table 5, assuming k = 3 and the objective is to maximize the objective function, after applying the three prey allocation methods, the prey position (e.g., $\vec{X}_{prey}$) will be; $\begin{pmatrix} 12 \\ 2 \end{pmatrix}$, $\begin{pmatrix} 7.667 \\ 6 \end{pmatrix}$, $\begin{pmatrix} 8.487 \\ 5.637 \end{pmatrix}$ for SLPA, MLPA, and WLPA respectively. The position of each leopard seal can be updated during the encircling phase by using (14)–(18). Initially, the k leaders are specified, and then the weight of each leader is calculated. After that, the location of the prey can be predicted using (11). Finally, the distance between the search agent and the potential prey should be calculated before the new position of each search agent is calculated using (14) [10,39]. After that, the new search agent's position is calculated.

$$\vec{D}^i_{L_m} = \left|\vec{C}.\vec{X}^i_{prey} - \vec{X}^i_{L_m}\right| \tag{14}$$

$$\vec{X}^{i+1}_{L_m} = \vec{X}^i_{prey} - \vec{A}.\vec{D}^i_{L_m} \tag{15}$$

$$\vec{A} = 2.\vec{a}.\vec{r}_1 - \vec{a} \tag{16}$$

$$\vec{C} = 2.\vec{r}_2 \tag{17}$$

$$a = 2 - i * \frac{2}{\mathrm{N_{Enc}}} \tag{18}$$

where $\vec{X}^i_{prey}$ is the predicted location of the prey at the iteration i, $\vec{X}^i_{L_m}$ is the mth leopard seal's position at iteration i, and $\vec{D}^i_{L_m}$ is the distance between the mth search agent and the prey. Also, $\vec{r}_1$ and $\vec{r}_2$ are random vectors $\in [0,1]$, $\vec{A}$ and $\vec{C}$ are coefficient vectors calculated using (16) and (17) [10,39], N_{Enc} is the iterations number in encircling phase using (6), and $\vec{a}$ vector decreases linearly from 2 to 0 over the course of iterations calculated using (18). During the encircle phase, $\vec{A}$ is set to a random value in $[-1,1]$, hence, the new position of the search agent will be anywhere in between the current

Table 6
Different iterations for the encircling phase of the illustrative example.

Agent	Initially			First Iteration (i = 1) $\vec{X}_{prey} = \begin{pmatrix}-3.126\\-3.304\end{pmatrix}$					Second Iteration (i = 2) $\vec{X}_{prey} = \begin{pmatrix}-3.256\\-3.592\end{pmatrix}$				
	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	Weight	$\vec{A}$	$\vec{C}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	Weight	$\vec{A}$	$\vec{C}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	Weight
L_1	$\begin{pmatrix}-2.792\\-0.999\end{pmatrix}$	−0.577	0	0.094	0.105	$\begin{pmatrix}-3.358\\-3.365\end{pmatrix}$	−5.877	0.835	0.060	1.922	$\begin{pmatrix}-3.43\\-3.804\end{pmatrix}$	−5.889	0
L_2	$\begin{pmatrix}-3.613\\-3.237\end{pmatrix}$	−5.338	0.882	0.872	1.234	$\begin{pmatrix}-3.339\\-4.037\end{pmatrix}$	−5.859	0.824	−0.199	0.886	$\begin{pmatrix}-3.165\\-3.422\end{pmatrix}$	−6.121	0.551
L_3	$\begin{pmatrix}-1.742\\-3.908\end{pmatrix}$	−4.125	0.657	0.295	1.701	$\begin{pmatrix}-4.181\\-3.809\end{pmatrix}$	−4.535	0	−0.294	0.671	$\begin{pmatrix}-2.669\\-3.181\end{pmatrix}$	−6.31	1
	$\begin{pmatrix}-2.913\\-4.544\end{pmatrix}$	−5.105	0.839	0.334	1.342	$\begin{pmatrix}-3.554\\-3.341\end{pmatrix}$	−5.552	0.633	0.796	1.042	$\begin{pmatrix}-3.384\\-3.912\end{pmatrix}$	−5.899	0.024
L_5	$\begin{pmatrix}-2.958\\-2.602\end{pmatrix}$	−5.500	0.912	0.0911	0.013	$\begin{pmatrix}-3.392\\-3.537\end{pmatrix}$	−5.914	0.858	0.001	0.413	$\begin{pmatrix}-3.258\\-3.594\end{pmatrix}$	−6.07	0.43
L_6	$\begin{pmatrix}-2.851\\-4.003\end{pmatrix}$	−5.974	1	−0.091	1.108	$\begin{pmatrix}-3.07\\-3.273\end{pmatrix}$	−6.142	1	−0.391	1.308	$\begin{pmatrix}-2.791\\-3.035\end{pmatrix}$	−6.191	0.717
$\vec{X}_{prey}$	$\begin{pmatrix}-3.126\\-3.304\end{pmatrix}$			$\vec{X}_{prey}$		$\begin{pmatrix}-3.256\\-3.592\end{pmatrix}$			$\vec{X}_{prey}$		$\begin{pmatrix}-3.031\\-3.3021\end{pmatrix}$		

Agent	Third Iteration (i = 3) $\vec{X}_{prey} = \begin{pmatrix}-3.031\\-3.3021\end{pmatrix}$				
	$\vec{A}$	$\vec{C}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	Weight
L_1	0.062	−0.099	$\begin{pmatrix}-3.062\\-3.335\end{pmatrix}$	−6.178	0.436
L_2	−0.2	−0.933	$\begin{pmatrix}-2.598\\-2.846\end{pmatrix}$	−6.125	0.209
L_3	0.031	−0.087	$\begin{pmatrix}-3.061\\-3.328\end{pmatrix}$	-6.176	1
			$\begin{pmatrix}-2.669\\-3.181\end{pmatrix}$	−6.31	
L_4	0.832	0.996	$\begin{pmatrix}-3.188\\-3.826\end{pmatrix}$	−6.076	0
L_5	0.061	0.963	$\begin{pmatrix}-3.208\\-3.496\end{pmatrix}$	−6.101	0.107
L_6	−0.098	−0.093	$\begin{pmatrix}-2.941\\-3.207\end{pmatrix}$	−6.208	0.564
	$\vec{X}_{prey}$		$\begin{pmatrix}-2.892\\-3.156\end{pmatrix}$		

Table 7
Best positions for all agents in illustrative example (after encircling).

Search agent	Position		Objective function	Rank
	x_1	x_2		
L_1	−3.062	−3.335	−6.178	3
L_2	−2.598	−2.846	−6.125	4
L_3	−2.669	−3.181	−6.31	1
L_4	−3.188	−3.826	−6.076	6
L_5	−3.208	−3.496	−6.101	5
L_6	−2.941	−3.207	−6.208	2

position of the search agent and the position of the prey. Table 6 shows the different iterations for the encircling phase of the illustrative example. In illustrative example (after encircling), the best positions for all leopard seal are presented in Table 7.

Leopard Seal Optimization (Encircling Phase)

Inputs:

- S: the set of *n* available suggested solutions or search agents (leopard seals).
- F(X): objective function to minimize or maximize.
- N: total number of iterations.
- K: number of Alpha individuals.
- The best positions of the search agents after the search phase.

Outputs:

- *The best positions of the search agents after the encircling phase.*

Steps:

Calculating the encircling number cycles

$N_{Enc} = \left\lfloor \frac{N}{3} \right\rfloor$

For i=1 To N_{Enc}

- Pick the k alpha agents

- Calculate the prey's position $\vec{X}^i_{prey}$ *assuming u dimension space as:*

$$\vec{X}^i_{prey} = \frac{1}{\sum_{m=1}^{k} W^i(L_m)} \begin{pmatrix} \sum_{m=1}^{k} W^i(L_m).x^i_{1m} \\ \sum_{i=1}^{k} W^i(L_m).x^i_{2m} \\ \cdots \\ \cdots \\ \cdots \\ \sum_{i=1}^{k} W^i(L_m).x^i_{um} \end{pmatrix}$$

Algorithm Parameters	
n	No. of search agents
P_m	The m[th] search agent
k	Number of alpha individuals
S	Set of available search agents
$\vec{X}^i_{prey}$	The current position of the prey at the it[h] iteration.
F(X)	Objective function
N	Total number of iterations
N_{Enc}	No. of encircling iterations.
i	Iteration number
$\vec{D}^i_{L_m}$	The distance vector from the m[th] agent to the prey.
$\vec{A}, \vec{C}$	Coefficient vectors
$\vec{r_1}, \vec{r_2}$	Random vectors ∈[0,1]

For each agent L_m *in S*

$a = 2 - i * \frac{2}{N_{Enc}}$

$\vec{A} = 2.\vec{a}.\vec{r_1} - \vec{a}$

$\vec{C} = 2.\vec{r_2}$

$\vec{D}^i_{L_m} = \left|\vec{C}.\vec{X}^i_{prey} - \vec{X}^i_{L_m}\right|$

$\vec{X}^{i+1}_{L_m} = \vec{X}^i_{prey} - \vec{A}.\vec{D}^i_{L_m}$

$Objective\_Value(L_m, i+1) = F(\vec{X}^{i+1}_{L_m})$

if $Objective\_Value(L_m, i+1) < Objective\_Value(L_m, i)$ *Then*

$\vec{X}^{i+1}_{L_m} = \vec{X}^i_{L_m}$

$Objective\_Value(L_m, i+1) = Objective\_Value(L_m, i)$

End if

Next

Next

6.3. Attacking the prey

After the completion of the prey encircling, the attack phase starts where the prey is completely unable to move and the search agents are very close to it. In the attack phase, the search agents try to get closer and closer to the prey so that they can capture it completely and then devour it. Therefore, the movement of the search agents in the attack phase is similar to the way they move in the encircling phase, with two differences, the first is that they try to reduce the distance between them and the prey as much as possible, so they can get more closer to it. The distance between the leopard seal and the potential prey during the attack phase can be calculated using (19).

$$\vec{D}^i_{L_m} = \left|\vec{X}^i_{prey} - \vec{X}^i_{L_m}\right| \tag{19}$$

where $\vec{X}^i_{prey}$ is the predicted location of the prey at the iteration i, $\vec{X}^i_{L_m}$ is the mth search agent's position at iteration i, and $\vec{D}^i_{L_m}$ is the distance between the mth leopard seal and the potential. The second difference is in the way the location of the potential prey is identified. During the attack phase, the used prey allocation method is SLPA. Hence, the prey position is the position of the search agent that maximizes the objective function validation, this search agent is called; "alpha agent". Then, $\vec{X}^i_{prey} = \vec{X}^i_{alpha}$, where $\vec{X}^i_{alpha}$ is the position vector of the alpha agent during the ith iteration. The next position of the search agent (e.g., $\vec{X}^{i+1}_{L_m}$) is calculated using (14).

Table 8
Different iterations for the attack phase of the illustrative example.

Agent	Initially		First Iteration (i = 1) $\vec{X}_{prey} = \begin{pmatrix}-2.669\\-3.181\end{pmatrix}$			Second Iteration (i = 2) $\vec{X}_{prey} = \begin{pmatrix}-2.681\\-3.239\end{pmatrix}$		
	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	$\vec{A}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	$\vec{A}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value
L_1	$\begin{pmatrix}-3.062\\-3.335\end{pmatrix}$	−6.178	−0.054	$\begin{pmatrix}-2.648\\-3.173\end{pmatrix}$	−6.31	−0.089	$\begin{pmatrix}-2.678\\-3.233\end{pmatrix}$	−6.322
L_2	$\begin{pmatrix}-2.598\\-2.846\end{pmatrix}$	−6.125	0.172	$\begin{pmatrix}-2.681\\-3.239\end{pmatrix}$	−6.323	0.972	$\begin{pmatrix}-2.681\\-3.239\end{pmatrix}$	−6.323
L_3	$\begin{pmatrix}-2.669\\-3.181\end{pmatrix}$	−6.31	0.0295	$\begin{pmatrix}-2.669\\-3.181\end{pmatrix}$	−6.31	0.761	$\begin{pmatrix}-2.691\\-3.282\end{pmatrix}$	−6.329
L_4	$\begin{pmatrix}-3.188\\-3.826\end{pmatrix}$	−6.076	0.334	$\begin{pmatrix}-2.842\\-3.396\end{pmatrix}$	−6.31	0.584	$\begin{pmatrix}-2.775\\-3.331\end{pmatrix}$	−6.321
L_5	$\begin{pmatrix}-3.208\\-3.496\end{pmatrix}$	−6.101	0.0911	$\begin{pmatrix}-2.718\\-3.21\end{pmatrix}$	−6.309	0.077	$\begin{pmatrix}-2.684\\-3.241\end{pmatrix}$	−6.323
L_6	$\begin{pmatrix}-2.941\\-3.207\end{pmatrix}$	−6.208	−0.591	$\begin{pmatrix}-2.508\\-3.166\end{pmatrix}$	−6.307	0.591	$\begin{pmatrix}-2.783\\-3.282\end{pmatrix}$	−6.311
$\vec{X}_{prey}$	$\begin{pmatrix}-2.669\\-3.181\end{pmatrix}$		$\vec{X}_{prey}$	$\begin{pmatrix}-2.681\\-3.239\end{pmatrix}$		$\vec{X}_{prey}$	$\begin{pmatrix}-2.691\\-3.282\end{pmatrix}$	

Agent	Third Iteration (i = 3) $\vec{X}_{prey} = \begin{pmatrix}-2.691\\-3.282\end{pmatrix}$			Fourth Iteration (i = 4) $\vec{X}_{prey} = \begin{pmatrix}-2.691\\-3.286\end{pmatrix}$			
	$\vec{A}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	$\vec{A}$	$\vec{X}_{Lm}$ $\begin{pmatrix}x_1\\x_2\end{pmatrix}$	Objective function value	
L_1	−0.8	$\begin{pmatrix}-2.681\\-3.243\end{pmatrix}$	−6.324	−0.098	$\begin{pmatrix}-2.69\\-3.282\end{pmatrix}$	−6.32889310	
L_2	0.078	$\begin{pmatrix}-2.691\\-3.286\end{pmatrix}$	−6.329	0.358	$\begin{pmatrix}-2.691\\-3.286\end{pmatrix}$	−6.329313603	
L_3	−0.191	$\begin{pmatrix}-2.691\\-3.282\end{pmatrix}$	−6.328	−0.701	$\begin{pmatrix}-2.691\\-3.283\end{pmatrix}$	−6.329071492	⟵Best Agent
L_4	0.714	$\begin{pmatrix}-2.751\\-3.317\end{pmatrix}$	−6.324	0.914	$\begin{pmatrix}-2.746\\-3.314\end{pmatrix}$	−6.325181631	
L_5	−0.665	$\begin{pmatrix}-2.686\\-3.255\end{pmatrix}$	−6.325	−0.001	$\begin{pmatrix}-2.691\\-3.286\end{pmatrix}$	−6.329310372	
L_6	−0.014	$\begin{pmatrix}-2.689\\-3.282\end{pmatrix}$	−6.328	−0.104	$\begin{pmatrix}-2.691\\-3.286\end{pmatrix}$	−6.32929124	
	$\vec{X}_{prey}$	$\begin{pmatrix}-2.691\\-3.286\end{pmatrix}$		$\vec{X}_{prey}$	$\begin{pmatrix}-2.691\\-3.286\end{pmatrix}$		

According to the illustrative example, considering the search agents L_1, L_2, …, L_6 presented in Table 8. For the attack phase (e.g., $N_{Att} = 4$), there are four iterations. Initially, based on the last information received from the encircling phase illustrated in Table 7, the best (alpha) search agent is L_3. The location vector for those alpha seal are ; $\vec{X}_{alpha} = \vec{X}_{L_3} = \begin{pmatrix}-2.669\\-3.181\end{pmatrix}$. This location is the virtual location of the prey. During the attack phase, Table 8 illustrates the needed calculations for the four consecutive iterations considering Eqs. (14)–(18). Related to the results in Table 8, the effectiveness of the performed exploitation is ensured during the attack phase because each leopard seal keeps providing better fitness value through the consecutive iterations. Accordingly, the effectiveness of LSO algorithm is improved by this behavior because it moves continuously towards the best or optimal solution. The steps of encircling phase for LSO are represented in algorithm 2.

6.4. The computational cost of LSO

In fact, the computational cost or complexity of LSO is influenced by three main factors. These factors are the population size (n), the number of maximum iterations (N), and the assessment evaluation function that represents the problem dimensions (ξ). According to population size factor, the complexity of population initialization process is $O(n)$. In this

Table 9
The main parameters of every optimization algorithm and their values through execution.

Algorithm	Parameter	Value
LSO	A number determine the shape of the movement of logarithmic spiral during the encircling phase (b)	1,2,3
	Number of alpha leopard seals (k) (the leaders of leopard seal through encircling and attacking phases)	4,7,11
TSO [32]	NP is the number of tuna populations	50
	$\alpha 1$ and $\alpha 2$ are weight coefficients that control the tendency of individuals to move towards the optimal individual and the previous individual	$\alpha 1 = a + (1 - a) \cdot t / t_{max}$ $\alpha 2 = (1 - a) - (1 - a) \cdot t / t_{max}$
	a is a constant used to determine the extent to which the tuna follow the optimal individual and the previous individual in the initial phase	0.7
	b is a random number	Random between [0,1]
	TF is a random number	1 or −1
	Each individual randomly chooses one of the two foraging strategies to execute, or chooses to regenerate the position in the search space according to probability (z)	0.05
	t_{max} is the maximum iterations	100
PSO [33]	m is the number of problem variables rand is a random number s is a random number R is a constant Cognitive and social constant (C1 , C2)	Random between [0,1] Random between [0,1] 1 or 2 0.2 (2, 2)
CMBO [34]	r is a random number	Random between [0,1]
AAM [35]	Random value	Random [0,1]
	SP shows the selection pressure	1.1
	$rand_n$ is a random number with normal distribution	
	$\alpha 1$ is a scaling parameter	From 0 to 1
	β is a random number with normal distribution	From 0 to 1
	$\alpha 2$ is a scaling parameter to adjust the movement step	In the range of (0,2]
	Pheromone which is initialized randomly in the beginning of the optimization process is used to control the ants' movement towards	In the range of [−1,1]
	b is a parameter to specify the shape and the angle of the flight and is selected randomly	In the range of (0,1]
	l which is selected randomly to control the closeness of the aphid in the spiral movement to the adjacent colony's ringleader	In the range of [−1,1]
WSO [36]	A is the acceleration factor which is constant From 2 to 0. r is a random number. c_1 and c_2 are two uniformly created random numbers τ denotes the acceleration coefficient r_1, r_2 and r_3 are random numbers	0.0005 Random [0,1] Random [0,1] 4.125 In the range of [0, 1]
RPO [38]	Number of Scouts (λ) in searching phase	10,15,20
	Number of alpha fishes (k) In encircling and attacking phases	5,7,10
	A number defines the shape of the movement logarithmic spiral during the encircling phase (b)	1,2,3
	Number of checkpoints (n_{ck}) in the case if there is a collision	3,7,9

work, the maximum iterations number N equals the summation of N_{Srch}, N_{Enc}, and N_{Att}. According to n, N, and ξ, the computational cost of LSO is measured using (20).

$$LSO's\ computational\ cost = O(n * N * \xi) \tag{20}$$

6.5. Statistical analysis of Leopard Seal Optimization (LSO) technique

In this subsection, the proposed LSO will be statistically tested against other six algorithms provided in Table 9. In fact, Table 9 shows the values of the applied parameters for every optimization algorithm that should be assigned through implementation. At maximum iterations number (N) = 400, these algorithms will be tested at four different numbers of search agents; $n = \{30, 60, 90, 120\}$ by using two statistical test methods called t-test (alpha = 0.05) [40] and Wilcoxon test ($p \geq 0.05$) as a non-parametric method [41]. Actually, these measurements are performed using 5 benchmark functions, which are; F1: Schwefel 2.22, F2: Schwefel 1.2, F3: Schwefel 2.21, F4: Griewank, and F5: Salomon [40] as presented in Table 10. According to Table 10, the 5 benchmark functions in this study have the same dimensions that equal 10; $f = 10$. Also, these benchmark functions have the same minimum fitness value that equals 0; $fit_{min} = 0$.

Two main statistical methods are used to test the performance of LSO against other 6 algorithms provided in Table 9 using five benchmark function provided in Table 10. These statistical methods are t-test using the significance level equals 0.05 (alpha = 0.05) [40] and Wilcoxon test as a non-parametric method using the significance level is greater than or equals 0.05 ($p \geq 0.05$) [41]. The results of measuring t-test and Wilcoxon rank-sum test for the used algorithms in Table 9 based on many different search agents numbers using 5 benchmark functions in Table 10 are presented in Table 11. These measurements are performed at 4 different search agents numbers equal 30, 60, 90, and 120.

According to Table 11, p-values are less than 0.05 in most cases except that the p-values for POA and LSO in F3 at search agents number = 30 and F4 at search agents number = 90 are greater than 0.05. This means that the optimization efficiency of all algorithms and LSO are significantly different if tested values are less than 0.05. On the other hand, the

Table 10
Five benchmark functions used in this study.

Label	Name	f	Domain [Lower, Upper]	Expression	fit_{min}
F1	Schwefel 2.22	10	[−10,10]	$F1(g)=\sum_{i=1}^{Dim}\lvert g_i\rvert+\prod_{i=1}^{Dim}\lvert g_i\rvert$	0
F2	Schwefel 1.2	10	[−100,100]	$F2(g)=\sum_{i=1}^{Dim}(\sum_{j-1}^{i} g_j)^2$	0
F3	Schwefel 2.21	10	[−100,100]	$F3(g)=\max_i\{\lvert g_i\rvert, 1\le i\le Dim\}$	0
F4	Griewank	10	[−600,600]	$F4(g)=\sum_{i=1}^{Dim}\frac{g_i^2}{4000}-\prod\cos\left(\frac{g_i}{\sqrt{i}}\right)+1$	0
F5	Salomon	10	[−100,100]	$F5(g)=1-\cos(2\pi)\sqrt{\sum_{i=1}^{Dim} g_i^2}+0.1\sqrt{\sum_{i=1}^{Dim} g_i^2}$	0

Table 11
t-test (alpha = 0.05) and Wilcoxon rank-sum test (p ≥ 0.05) measurements for LSO against other six algorithms at N = 400.

Algorithm		Function											
		TSO		POA		CMBO		AAM		WSO		RPO	
		t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value
F1	30	0.005	$3.02*10^{-12}$	0.006	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.013	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$
	60	0.01	$4.02*10^{-12}$	0.009	$3.05*10^{-11}$	0.013	$5.02*10^{-10}$	0.001	$3.32*10^{-12}$	0.002	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$
	90	0.011	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$	0.013	$3.32*10^{-12}$
	120	0.002	$3.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.006	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$
F2	30	0.008	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.009	$5.02*10^{-10}$	0.01	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$
	60	0.001	$5.02*10^{-10}$	0.011	$4.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.009	$5.02*10^{-12}$	0.005	$3.02*10^{-12}$
	90	0.009	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$	0.009	$5.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.011	$2.02*10^{-12}$	0.01	$3.02*10^{-12}$
	120	0.013	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.013	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$
F3	30	0.002	$3.02*10^{-12}$	**0.302**	$\mathbf{1.03*10^{-1}}$	0.013	$3.02*10^{-12}$	0.01	$3.32*10^{-12}$	0.002	$5.02*10^{-10}$	0.011	$5.02*10^{-10}$
	60	0.006	$3.32*10^{-12}$	0.013	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$	0.003	$3.02*10^{-12}$	0.006	$3.02*10^{-12}$
	90	0.01	$3.02*10^{-12}$	0.002	$3.32*10^{-12}$	0.006	$3.32*10^{-12}$	0.011	$3.02*10^{-12}$	0.004	$3.02*10^{-12}$	0.005	$5.02*10^{-10}$
	120	0.013	$3.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.013	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$
F4	30	0.002	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$	0.005	$4.02*10^{-12}$	0.009	$3.32*10^{-12}$	0.009	$3.02*10^{-12}$
	60	0.001	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$	0.013	$5.02*10^{-10}$
	90	0.009	$5.02*10^{-10}$	0.011	$\mathbf{3.12*10^{-1}}$	0.002	$4.02*10^{-12}$	0.013	$3.02*10^{-12}$	0.006	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$
	120	0.011	$3.02*10^{-12}$	0.013	$3.02*10^{-12}$	0.001	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.001	$4.02*10^{-12}$
F5	30	0.006	$3.02*10^{-12}$	0.002	$4.02*10^{-12}$	0.005	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$	0.013	$4.02*10^{-12}$	0.005	$3.02*10^{-12}$
	60	0.009	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$	0.006	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.01	$3.02*10^{-12}$
	90	0.002	$3.32*10^{-12}$	0.013	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.005	$4.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.009	$3.02*10^{-12}$
	120	**0.082**	$3.02*10^{-12}$	0.002	$3.02*10^{-12}$	0.004	$4.02*10^{-12}$	0.01	$3.02*10^{-12}$	**0.103**	$3.02*10^{-12}$	0.011	$3.02*10^{-12}$

optimization efficiency of POA and LSO is similar in F3 at search agents number = 30 and F4 at search agents number = 90 because tested values are greater than 0.05. According to *t*-test values in Table 11, it is noted that the optimization efficiency of all algorithms and LSO are significantly different in most cases where tested values are less than 0.05. On the other hand, the optimization efficiency of POA and LSO is similar in F3 at search agents number = 30 that is the same result of Wilcoxon test. Additionally, *t*-test measurements introduced that the optimization efficiency of TSO and LSO is similar in F5 at search agents number = 120 and also the optimization efficiency of WSO and LSO is similar in F5 at search agents number = 120 because tested values are greater than 0.05. All values that are greater than 0.05 have been highlighted in Table 11. Based on these results, the final decision on the proposed LSO compared to other recent algorithms will be accurately taken from Wilcoxon test as a non-parametric test method more than *t*-test as a parametric method. Accordingly, statistical analysis shows that the proposed LSO can provide fast and more accurate solutions compared to most recent algorithms and also it can provide solutions closer to optimal solutions.

7. Case study: Feature selection using Leopard Seal Optimization (LSO)

When constructing a predictive model, feature selection is a very important process to minimize the extracted features number by choosing a subset of consistent and related features. As a result, selecting the optimal features will minimize modeling computational costs while also enhancing model performance. It also decreases the storage requirement, and increases the convergence of learning. The underlying idea of feature selection is that the input data can contain numerous features that are either unnecessary or redundant and can thus be deleted with no loss of information. Fig. 8 depicts the benefits of feature selection.

Fig. 9 depicts the broad framework used to test the efficiency of the suggested LSO. This framework has two primary phases; the Phase of Feature Selection (PFS) and (ii) the Phase of classification (PC). The proposed LSO's effectiveness will be evaluated in PFS by identifying the most important features to detect Covid-19. The training dataset for PFS is a set of blood tests collected from infected people with Covid-19 and other tests for healthy people. As shown in Fig. 9, the most

Fig. 8. Benefits of feature selection.

significant features for Covid-19 diagnoses will be chosen during PFS by applying the introduced LSO. Next, in PC, Naïve Bayes (NB) classifier will be learned depending on the valid dataset without irrelevant features to accurately diagnose patients [42,43].

In order to classify a new case either to "Infected" or "Healthy" classes. In fact, it is essential to say that the LSO's version used to solve the feature selection problem is named Binary Leopard Seal Optimization (BLSO). According to BLSO, each leopard seal that represents a solution is expressed by a binary vector $L = (L_1, L_2, \ldots, L_f)$, $L_j \in [0,1]$ in f-dimensional space. Actually, each feature is represented in one of these dimensions. Thus, in the jth dimension, "0" value refers to that the corresponding jth feature is not selected, but "1" value refers to that the jth feature is selected. The binary representation of leopard seal solution is shown in Fig. 10.

As illustrated in Fig. 9, there are numerous consecutive steps to execute the BLSO to choose the optimal set of features from the Covid-19 dataset. According to Fig. 11, BLSO starts with a Swarm (S) that has a collection of leopard seals represented by L. For illustration, if S consists of n search agents, i.e. S contains $L = \{L_1, L_2, \ldots, L_h, \ldots, L_n\}$. Each leopard seal ($L_m$) in S provides a potential solution (includes a set of the best features) in the f-dimensional agent, the maximum features number in the Covid-19 dataset is denoted by f, and thus the mth leopard seal can be described as; $L_m = (L^1{}_m, L^2{}_m, \ldots, L^f{}_m)$. Each leopard seal's position (bit) value (L_m) is a binary value that could be zero or one. At first, S will be randomly generated to include n of search agents in a binary space. Then, the accuracy of NB model will be calculated as a fitness function to evaluate all agents in S using (21) [44].

$$Fitness\,(L_m) = NB\_Accuracy\,(L_m) \tag{21}$$

where $Fitness\,(L_m)$ is objective value of the mth leopard seal and $NB\_Accuracy(L_m)$ is the accuracy of NB classifier based on the selected features represented in the mth leopard seal. Related to the evaluation values, the best leopard seal (solution) is the one that provides the highest accuracy value. Hence, the fundamental goal of the feature selection procedure is to maximize the classifier's accuracy. Following the generation of the initial swarm of leopard seal utilizing n leopard seal in a binary space, the leopard seals are tested using NB model's accuracy. The number of position vectors of the leopard seal through its movement in each iteration (ξ), as well as the number of iterations (N) should be assigned. Based on the N value, the iterations number for the three phases called searching, encircling, and attacking will be computed by applying (5)–(7) respectively. According to N_{Srch}, N_{Enc}, and N_{Att}, the procedures to implement the suggested BLSO in Fig. 11 are arranged into three parts; (i) searching phase, (ii) encircling phase, and (iii) attacking phase. The searching phase steps will be performed initially until the iterations number (N_{Srch}) is reached. If the N_{Srch} is not terminated, the position vectors of each agent will be modified by (1), and the fitness function will be calculated. Then, these positions must be converted to a binary value by employing a sigmoid function (22) [44].

$$L^j_{binary\_m}\,(t+1) = \begin{cases} 1 & \text{if rand } (0,1) \geq \text{sigmoid}(L^j_m) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where the binary value of mth leopard seal at jth bit in the following iteration $t+1$ is $L^j{}_{binary\_m}\,(t+1)$; $j = 1,2,3,\ldots,f$, and a value between [0,1] is randomly generated by $rand(0,1)$. Furthermore, the probability of jth bit taking 1 or 0 value called the sigmoid function is $sigmoid(L^j{}_m)$that can be computed using (23) [44].

$$\text{sigmoid}\left(L^j_m\right) = \frac{1}{1+e^{-L^j_m}} \tag{23}$$

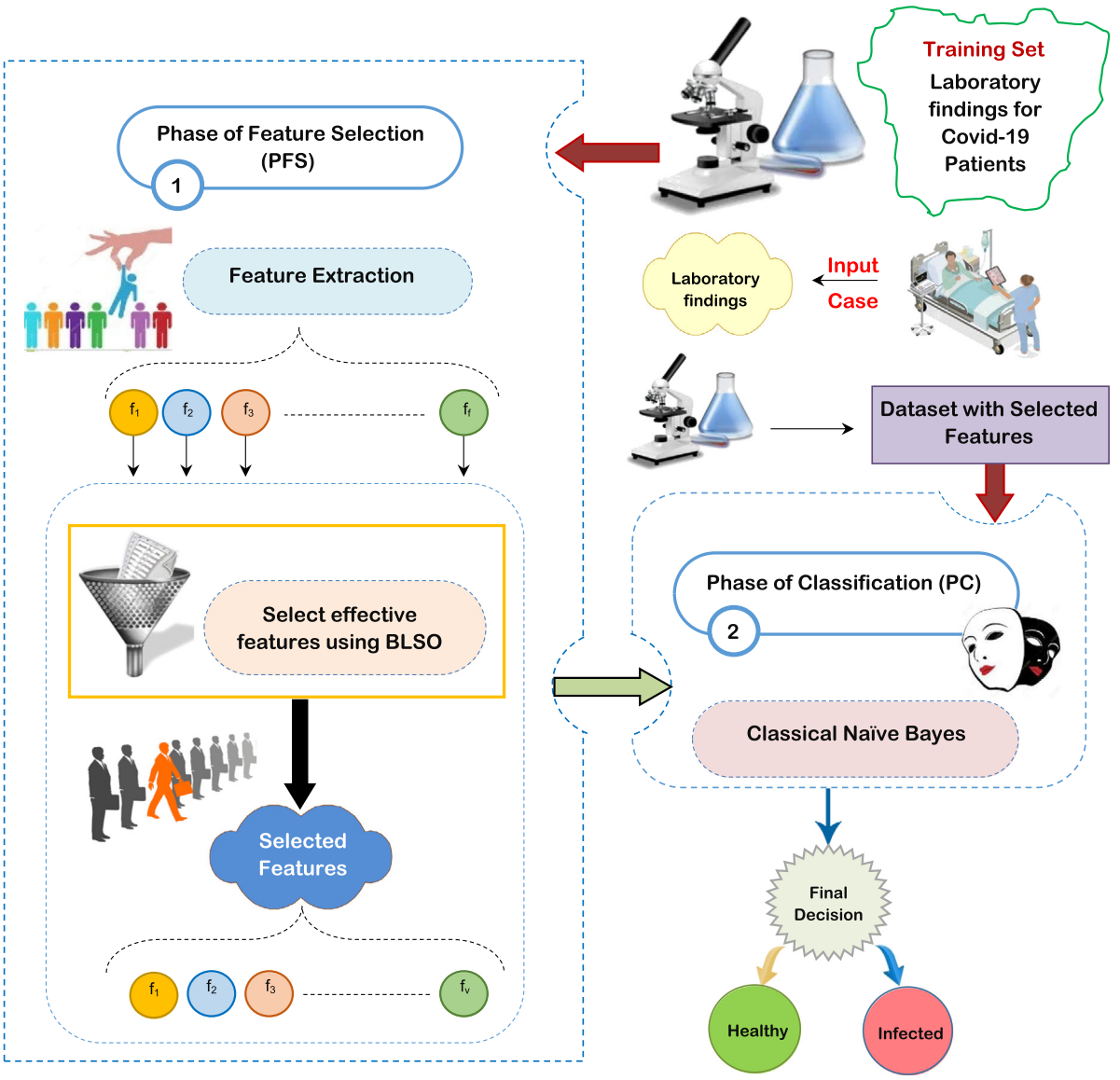


Fig. 9. The employed framework to test the efficiency of the proposed BLSO.

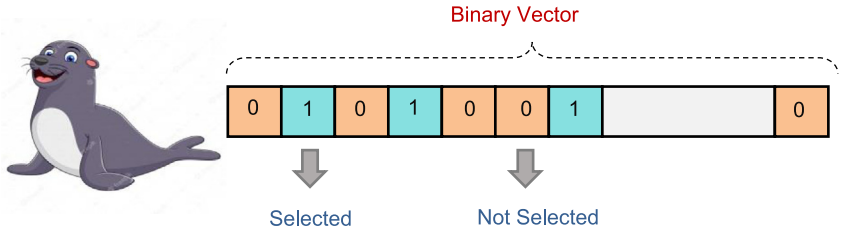


Fig. 10. Binary representation for the Leopard Seal.

where e is the base of the natural logarithm. Each search agent in S based on the new position $L^j_{binary\_m}\ (t+1)$ will be tested using (21). During their trip in seeking of prey, each leopard in S will keep modified position and evaluation values. The steps of the searching phase will be repeated till the N_{Srch} is finished. After completing the searching phase, the best position in S for each leopard will be the one that gives the highest fitness value through their travel in pursuit of prey.

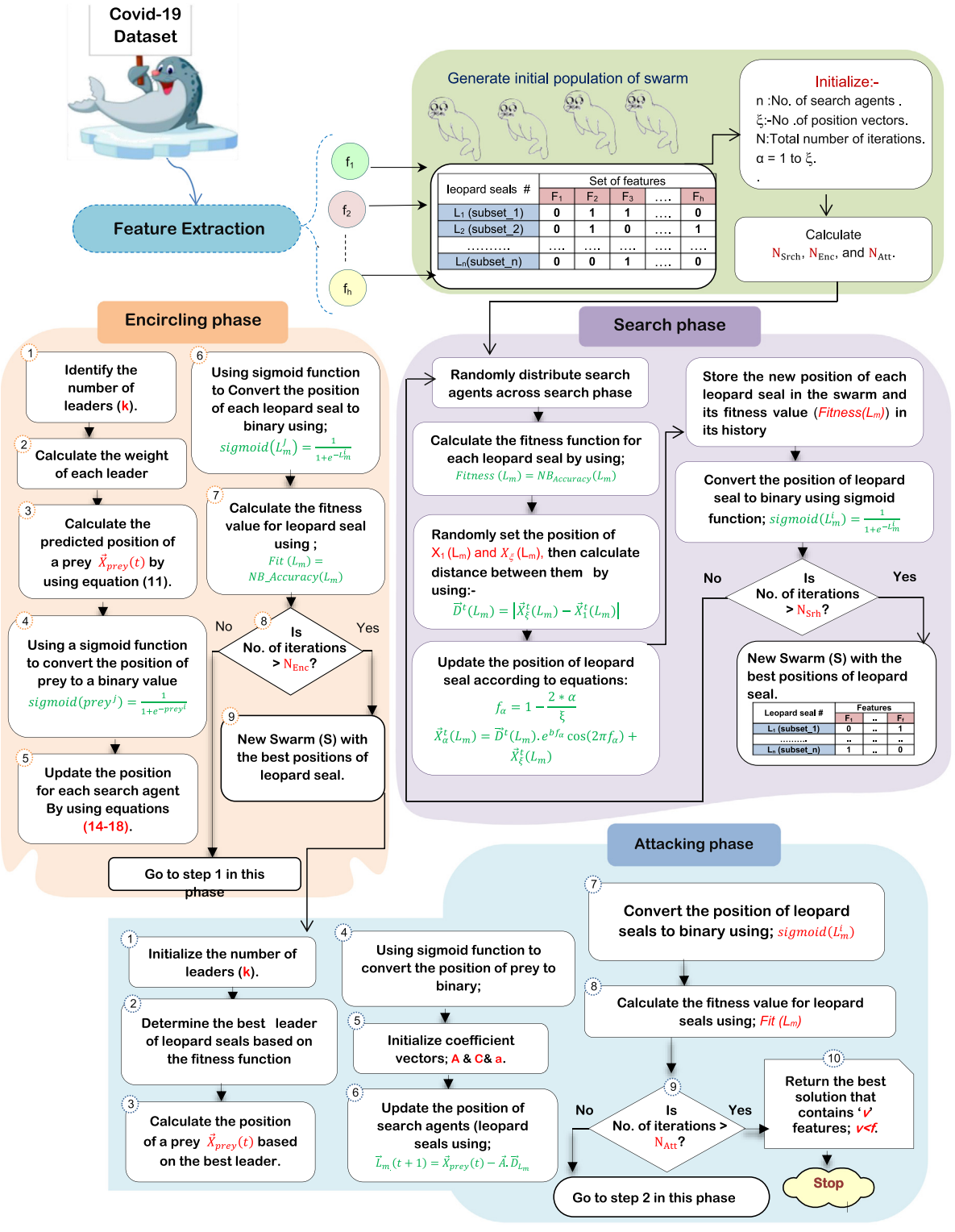


Fig. 11. Steps of implementing the BLSO.

The procedures of the encircling phase will be executed based on the best positions of the leopard seals (search agents) in S provided by the searching phase until the number of iterations (N_{Enc}) is finished. Firstly, identify the best leaders

Table 12
The common parameters and their values through implementation for all used optimization algorithms.

Parameter	Description	Assigned Value
N	The iterations number	40, 80, 120, 160, 200, 240, 280, 320, 360,400
n	The search agents number	30, 60, 90, and 120
f	Number of features used to detect Covid-19 (dimensions)	110
v	Number of selected features (after optimization)	Determined based on the employed algorithm
rand	random value used in the sigmoid function	Random between [0,1]

according to the calculations of fitness function, then using WLPA which is a new method for prey allocation that reliably predicts the location of the prey. According to WLPA, each of the selected leaders is given a weight depending on its closeness to the prey, which can be measured by the extent to which this search agent accomplishes the objective function. The prey's position will be determined according to the positions of alpha leopard seals (k) using (11). If the N_{Enc} is not completed, the search agents will change their positions depend on the position of the prey using (14)–(18). At the end of this phase, the new position of each leopard in S is computed by initially measuring the distance between the search agent and the target prey using (14), then the new placement of the agent is computed using (15) and evaluated by fitness function to select the best leaders. All positions must be represented by binary values by using the sigmoid function in (23).

The attacking phase is beginning based on the last information received from the encircling phase. Firstly, determine the best (alpha) search agents and use their position in order to determine where the target prey $\vec{X}_{prey}(t)$ is located. As mentioned above, the sigmoid function will be used to convert the values of positions to binary numbers. While N_{Att} is not reached, the search agents will update their position and evaluate the fitness function according to this. Finally, the fittest leopard seal is the optimal solution with the optimal set of features. It should be emphasized that iterations divided into searching, encircling, and attacking phases give the BLSO algorithm the ability to produce a rapid and precise set of features. The main cause for this is that the technique allows each phase to try numerous times to find the optimal answer before executing the next step to catch the prey, which allows the next phases to be executed more rapidly and correctly.

7.1. The used dataset and employed parameters

The dataset utilized to compare the proposed LSO algorithm to the recent and popular optimization techniques will be discussed in this subsection. Then, the values of the parameters which used during the execution will be explained. The Albert Einstein dataset is used in this work. This dataset was gathered from the Albert Einstein Hospital and provided through Kaggle [45]. This dataset includes 5644 cases collected between March 28, 2020 and April 3, 2020. This valuable dataset contains 110 features and covers numerous clinical tests such as blood, urine, RT-PCR, and other tests of Covid-19.

In this dataset, there are two main class types termed "positive" as well as "negative". In fact, the positive class relates to people who are infected with Covid-19 but the negative class indicates to people who are not infected with Covid-19. The used dataset contains 559 positive patients and 5085 negative ones [45]. In fact, the suggested BLSO method will be applied on the used dataset to identify the features that have the greatest effect on Covid-19 diagnosis. To find the best subset of features, the performance of BLSO will be compared to six optimization techniques which represents the most recently SIA algorithms.

These six optimization techniques are; Tuna Swarm Optimization (TSO) [32], Pelican Optimization Algorithm (POA) [33], Cat and Mouse-Based Optimization (CMBO) [34], Aphid–Ant Mutualism (AAM) [35], White Shark Optimizer (WSO) [36], and Red Piranha Optimization (RPO) [38]. As a result, Table 12 illustrates the parameters as well as their values for all used techniques, which are; the search agents number (n) includes 30, 60, 90, and 120, (ii) the total iterations number (N) includes 40, 80, 120, 160, 200, 240, 280, 320, 360, and 400, and (iii) each agents' dimensions (f) = 110. In the sigmoid function, the random value is a value that belongs to the range between (0,1) used to transform continuous locations to binary. It is necessary to test the efficiency of the suggested BLSO against the optimization algorithms using these parameters values to establish the optimal methodology as a feature selection algorithm. As a result, the assessment measures will be described in the following subsection based on Table 9 that shows the applied parameters values for every optimization algorithm that should be used through implementation.

7.2. Evaluation metrics

Two main assessment measures called accuracy (fitness value) and execution time are used to evaluate all optimization techniques. The accuracy as well as running or execution time are applied to identify the optimal technique that can efficiently determine the optimal set of features in the dataset. Actually, the optimal technique can give the highest accuracy value at the least amount of running time compared to other techniques. To compute the every optimization technique's accuracy, the Covid-19 dataset is split into two parts called training and testing. Then, the NB classifier will be trained based on training data including the determined features by each technique and then it will be tested based on

Table 13
The characteristics of simulation platform.

Hardware	Software
Laptop @2.11G with 16.0 GB of RAM	MATLAB 2018a Windows 10 (64 bit) operating system Intel (R) Core (TM) i5-10210U

testing data [42]. At the end, the confusion matrix will be applied to compute the accuracy of NB for every algorithm [44]. In fact, the accuracy of *k*th technique using NB method can be computed by (24) [38].

$$Accuracy\ (optimization\_algorithm_k) = Max(NB\_accuracy(L_m)) \tag{24}$$

where *Accuracy* ($optimization\_algorithm_k$) is the accuracy of *k*th technique; k = 1,2,3…,10. $Max(NB\_accuracy(L_m))$ is the maximum NB accuracy value given by the best search agent L_m where $m = 1,2,3,\ldots,n$ and n is the total number of search agents. The accuracy of *m*th search agent in the swarm can be computed using (25) [38,44].

$$NB\_accuracy(L_m) = \frac{the\ number\ of\ correct\ classifications}{total\ number\ of\ classifications} * 100 \tag{25}$$

The second metric represents the running time that equals the multiplication of the leopard seals number by the iterations number in (26) [38].

$$Execution\ time\ (optimization\_algorithm_k) = n * N \tag{26}$$

where *time* ($optimization\_algorithm_k$), is the running time of the *k*th technique, N is the iterations number that relies on the stop criteria ($(optimization\_algorithm_k(t+1)) < 10^{-7}$), and n is the search agents number. Also, t is the current iteration number while the next iteration number is $t + 1$. Hence, N may include the iterations number (N_o) that is smaller than the maximum iterations number; $N = N_o$ in the case if the stopping criteria is met; else, it equals the total iterations number. In the next subsection, the experimental findings will be analyzed.

7.3. Experimental results

In this section, the experimental results to measure the performance of BLSO over the other recent SIA algorithms called TSO [32], POA [33], CMBO [34], AAM [35], WSO [36], and RPO [38] are provided during two situations. These situations are (i) testing the efficiency of BLSO based on the search agents number as well as the iterations number and (ii) giving a comparison study to test the results of BLSO against the other techniques. In the 1st situation, the accuracy as well as running time of BLSO are measured by using Eqs. (25) and (26). These measurements will be determined based on different search agents numbers (n), and different iterations numbers (N) as presented in Table 12. In the 2nd situation, a comparison study among BLSO and the other six algorithms will be presented by working out the accuracy as well as running time of them utilizing (25) and (26) respectively. These computations will be performed by different iterations number (N) at each predefined search agents number independently. Truth be told, the six techniques are implemented as a wrapper techniques to choose the most informative features that gives NB the ability to rapidly and accurately diagnose cases. Our execution utilizes the dataset called Albert Einstein to choose the optimal features among 110 features which essentially affect Covid-19 patients [45].

According to simulation platform, the experiments were performed on the same platform where all algorithms were tested and compared in MATLAB 2018a that was installed on a laptop that have Windows 10 (64 bit) operating system and Intel (R) Core (TM) i5-10210U. Also, this laptop have @2.11G with 16.0 GB of RAM. These characteristics of simulation platform describing both hardware and software are presented in Table 13.

7.3.1. Evaluating the performance of Binary Leopard Seal Optimization (BLSO) technique

The experimental findings for determining the efficiency of BLSO will be supplied in this subsection. BLSO is used as a wrapper feature selection method to choose the optimal features among 110 features that have a substantial effect on Covid-19 cases in quickly and properly manner. In reality, the experimental findings will be obtained by using two primary performance metrics, accuracy and execution time. The accuracy and execution time of BLSO are determined for different iterations numbers (N) and different search agents numbers (n), as illustrated in Figs. 12 and 13. By using Eqs. (25) and (26), the accuracy as well as running time of BLSO will be calculated.

As previously stated, Figs. 12 and 13 depict the effectiveness of the BLSO technique. As a result, Figs. 12 and 13 indicate that the accuracy as well as execution time of the BLSO technique increase progressively as both the iterations number (N) and the search agents number (n) increase. It is worth noting that the highest accuracy values are obtained when the iterations number (N) = 400, which are 95.03%, 95.33%, 96.9%, and 97.62% when the search agents number equals 30, 60, 90, and 120 respectively. Furthermore, when the number of iterations equals 40, the lowest execution time values are generated and the values are 867, 921, 1001, and 1314 at leopard seals number equals 30, 60, 90, and 120 respectively. Hence, it is proved that BLSO is a simple and flexible algorithm that can deal with feature selection problem as a real time problem introducing more accurate solutions related to different numbers of leopard seals and iterations. In the next subsection, BLSO algorithm will be tested against other recent algorithms at many different iterations number and many different of leopard seals numbers as provided in Table 12. These conditions are common for all optimization algorithms.

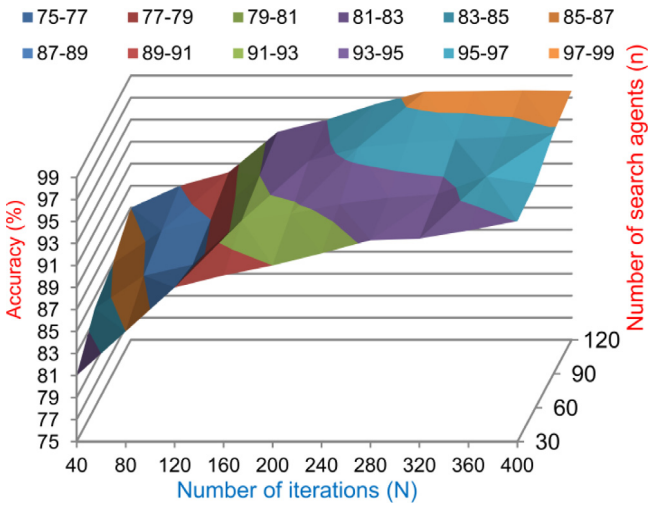


Fig. 12. The accuracy of BLSO.

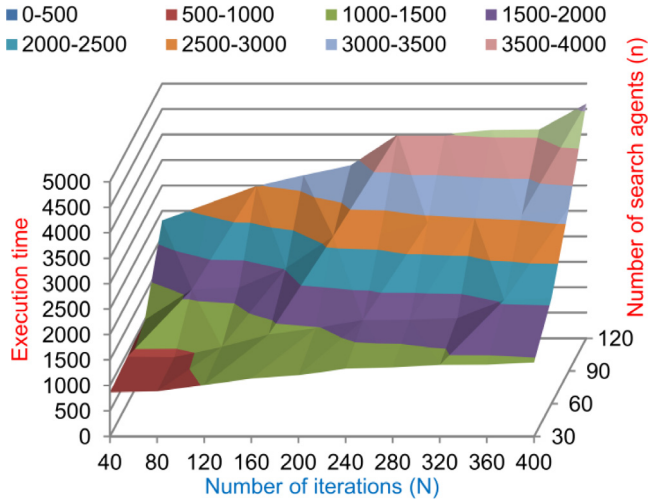


Fig. 13. The execution time of BLSO.

7.3.2. Comparing the efficiency of BLSO against recent competitors

The performance of the BLSO technique is measured in this comparative study to illustrate its effectiveness in comparison to the performance of the other recent six techniques provided in Table 9. In practise, the accuracy as well as execution time shown in (25) and (26) are utilized as evaluation metrics to compare the efficiency of BLSO to other algorithms based on the number of iterations (N). These measurements are carried out independently for each number of leopard seals (30, 60, 90, and 120). The BLSO is running based on these leopard seals numbers, and the accuracy as well as execution time measurements are calculated independently for each number of leopard seals (n) as illustrated in Figs. 14→21. Tables 14→17 depict the values of the maximum accuracy values and the minimum running time values for different algorithms at different leopard seals numbers.

As illustrated in Figs. 14→21, the accuracy as well as execution time of all techniques increases with the increment of the leopard seals number (n) and the iterations number (N). The reason is that the greater the leopard seals number and the iterations number, the greater the chance of reaching to the global optimal solution than reaching to the local optimal solution, and this takes more time to reach. Fig. 14 depicts the accuracy of all algorithms based on 30 search agents and the number of iterations (N). The results show that the BLSO offers the highest (best) value of accuracy while the POA delivers the lowest (worst) value throughout all iterations that is because POA suffers from low accuracy and high convergence time. At the maximum number of iterations (N = 400), the best accuracy values of TSO, POA, CMBO, AAM, WSO, RPO, and BLSO are 83.5%, 82.4%, 83.05%, 85.6%, 86%, 92.3% and 95.03%.

Fig. 15 shows that the BLSO gives the shortest execution time value for different iterations number. At the minimum iterations number (N) = 40, the minimum running time values of TSO, POA, CMBO, AAM, WSO, RPO, and BLSO are 1680,

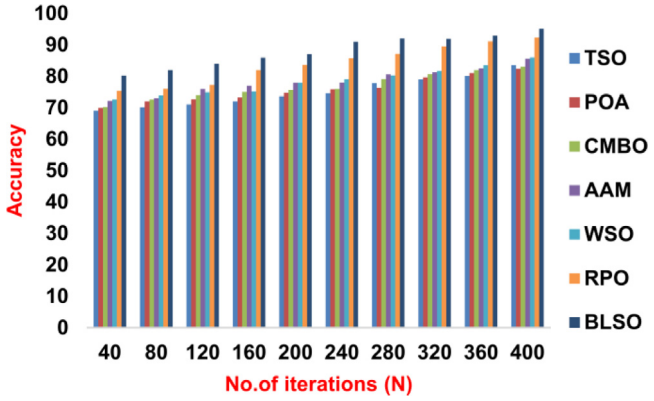


Fig. 14. Accuracy for different optimization techniques at search agent = 30.

Table 14
Values of accuracy and execution time for different optimization techniques at search agent = 30.

Number of search agent = 30		
Optimization technique	Accuracy at N = 400	Execution time at N = 40
TSO	83.5	1680
POA	82.4	1540
CMBO	83.05	1032
AAM	85.6	1080
WSO	86	958
RPO	92.3	917
BLSO	95.03	867

Table 15
Values of accuracy and execution time for different optimization techniques at search agent = 60.

Number of search agent = 60		
Optimization technique	Accuracy at N = 400	Execution time at N = 40
TSO	84.01	1776
POA	83.5	1810
CMBO	84.2	1117
AAM	86.3	1116
WSO	88.9	1050
RPO	93.09	993
BLSO	95.33	921

1540, 1032, 1080, 958, 917, and 867 respectively. It should be noted that the optimal (highest) accuracy values of all techniques are offered at the maximum iterations number (N) = 400, but their optimal (lowest) execution time values are given at the smallest iterations number (N) = 40. Although all techniques give the maximum running time values at N = 400, the running time of BLSO is lower than other algorithms. Thus, at the leopard seals number (n) = 30, results showed that BLSO outperforms other algorithms because it can provide fast and more accurate solutions compared to other algorithms. Additionally, RPO is the second best algorithm after BLSO because it depends on fewer parameters and it is a simple and flexible algorithm but POA is the worst algorithm because it cannot reach to global optimal solution as shown in Figs. 14, 15, and Table 14.

The accuracy values of all techniques based on 60 search agents are presented in Fig. 16 while their execution duration are provided in Fig. 17. Compared to the results at n = 30, the accuracy as well as execution time values of all techniques are increased at n = 60. The BLSO provides the highest accuracy values, while the POA introduces the lowest accuracy values over all iterations. At N = 400, the greatest accuracy values for TSO, POA, CMBO, AAM, WSO, RPO, and BLSO are 84.01%, 83.5%, 84.2%, 86.3%, 88.9%, 93.09% and 95.33% respectively. As demonstrated in Fig. 17, the BLSO provides the shortest execution time values whereas POA provides the maximum values according to all iterations, with values reaching 921 and 1810 respectively when the iteration number (N) = 40. The running time of TSO, POA, CMBO, AAM, WSO, RPO, and BLSO are 1776, 1810, 1117, 1116, 1050, 993, and 921 respectively. Similar to the results at n = 30, the best second and third algorithm after BLSO are RPO and WSO respectively while the worst algorithm is POA as shown in Figs. 16, 17, and Table 15.

Figs. 18 and 19 and Table 16 show the accuracy as well as running time values for all techniques when n = 90. At n = 90, the accuracy as well as running time of all techniques rise more than the values at n = 30 and n = 60. Fig. 18 shows that the BLSO gives the highest accuracy values at all numbers of iterations while the POA provides the lowest accuracy value when N = 400. The highest accuracy values for TSO, POA, CMBO, AAM, WSO, RPO, and BLSO at N = 400 are 84.7%, 83.8%, 84.25%, 86.66%, 88.989%, 94.62%, and 96.9% respectively. At n = 90 and N = 40, the execution time values of TSO, POA, CMBO, AAM, WSO, RPO, and BLSO are 1692, 1870, 1350, 1165, 1147, 1124, and 1001 as illustrated in Fig. 19. Similar

Table 16
Values of accuracy and execution time for different optimization techniques at search agent = 90.

Number of search agent = 90		
Optimization technique	Accuracy at N = 400	Execution time at N = 40
TSO	84.7	1692
POA	83.8	1870
CMBO	84.25	1350
AAM	86.66	1165
WSO	88.989	1147
RPO	94.62	1124
BLSO	96.9	1001

Table 17
Values of accuracy and execution time for different optimization techniques at search agent = 120.

Number of search agent =120		
Optimization technique	Accuracy at N = 400	Execution time at N = 40
TSO	85.3	2070
POA	84.02	1910
CMBO	86.09	1487
AAM	87.8	1530
WSO	90.24	1469
RPO	95.73	1409
BLSO	97.62	1314

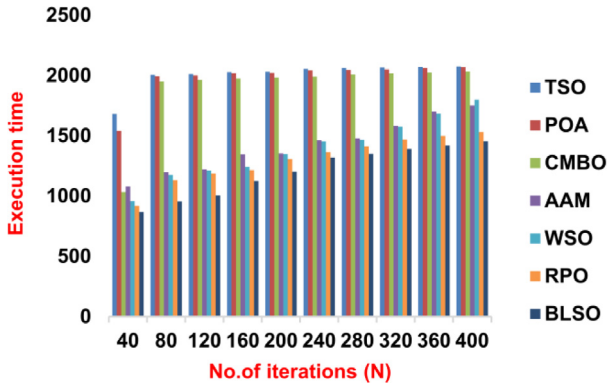


Fig. 15. Execution time for different optimization techniques at search agent = 30.

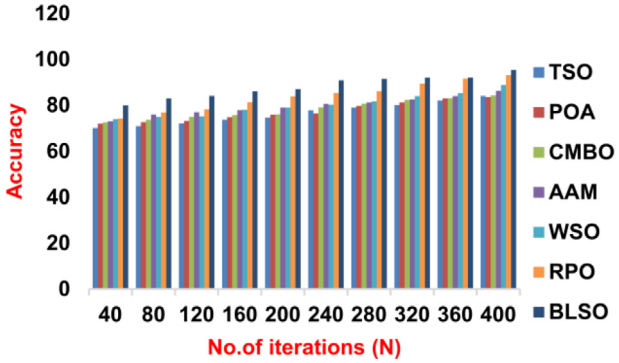


Fig. 16. Accuracy for different optimization techniques at search agent = 60.

to the results at n = 30 and n = 60, the best second algorithm after BLSO is WSO while the worst algorithm is POA as shown in Figs. 18, 19, and Table 17. When n = 120, the accuracy values of TSO, POA, CMBO, AAM, WSO, RPO, and BLSO are 85.3%, 84.02%, 86.09%, 87.8%, 90.24%, 95.73, and 97.62% respectively as illustrated in Fig. 20. Thus, BLSO represents the best algorithm that can provide the maximum accuracy values while POA is the worst algorithm because it provides the minimum accuracy values as shown in Fig. 20. In Fig. 21, the running time values of the same algorithms in the same order are 2070, 1910, 1487, 1530, 1469, 1409 and 1314 at n = 120. Accordingly, related to the results at n = 120, the shortest execution time algorithm is BLSO while the longest execution time is TSO. When the BLSO algorithm is compared to other algorithms, it is found that the BLSO provides the highest accuracy values and also providing the shortest execution time

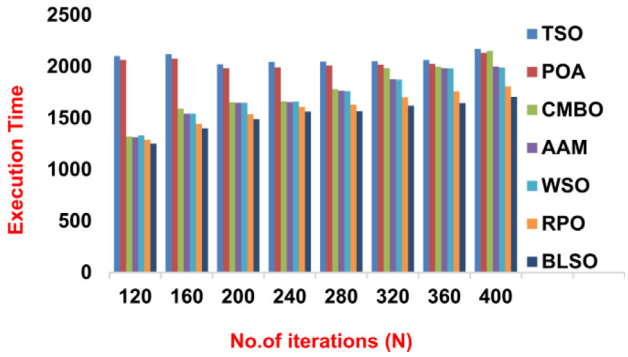


Fig. 17. Execution time for different optimization techniques at search agent = 60.

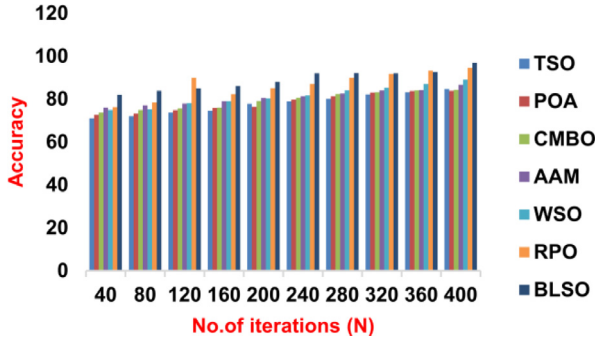


Fig. 18. Accuracy for different optimization techniques at search agent = 90.

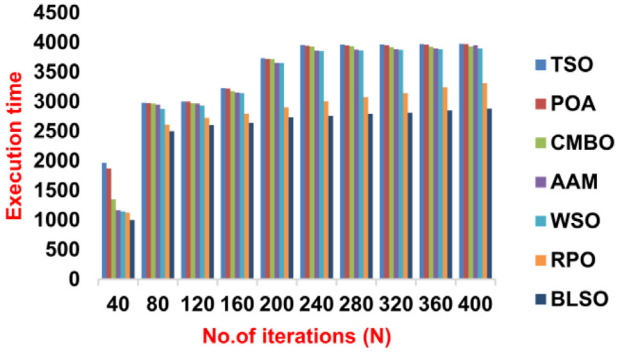


Fig. 19. Execution time for different optimization techniques at search agent = 90.

values. Based on the results at all values of n; n = {30,60,90,120}, the best algorithm is BLSO, the second best one is RPO, and the third best one is WSO while the worst one is POA.

8. Discussion of results

In fact, statistical analysis for the proposed LSO algorithm compared to other recent algorithms based on Wilcoxon test and *t*-test proved that LSO is superior than TSO, POA, CMBO, AAM, WSO, and RPO in most cases using common five benchmark functions indicated by {F1, F2, F3, F4, and F5} and four numbers of search agents n = {30,60,90,120}. The reason of superiority the LSO is that it has different characteristics compared to other algorithms where tested values are less than 0.05. As shown in Table 11, both Wilcoxon test and *t*-test proved that the effectiveness of POA and LSO is similar in F3 when n = 30 because the tested value is greater than 0.05. Wilcoxon test proved that POA and LSO is similar in F4 at n = 90 but *t*-test did not prove this. On the other hand, *t*-test measurements introduced that the optimization efficiency of TSO and LSO is similar in F5 at n = 120 and also the optimization efficiency of WSO and LSO is similar in F5 at n = 120 but Wilcoxon test did not prove this.

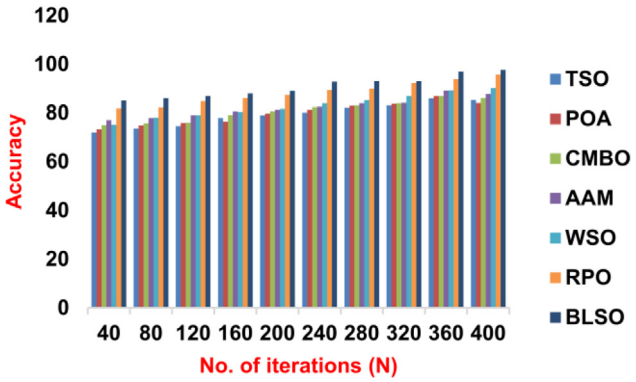


Fig. 20. Accuracy for different optimization techniques at search agent = 120.

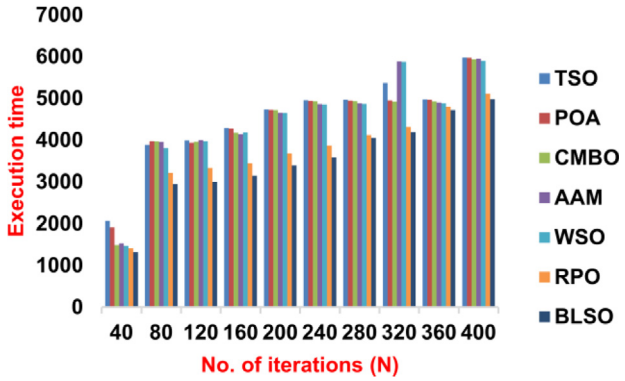


Fig. 21. Execution time for different optimization techniques at search agent = 120.

Based on Figs. 14→21 and Tables 14→17, it is summarized that

- The accuracy as well as execution time of all techniques increase with the increment of the search agents number (n) and the iterations number (N).
- At all numbers of search agents n = {30,60,90,120}, the maximum accuracy values are provided at the maximum number of iteration (N = 400) while the minimum execution time values are provided at the minimum iterations number (N = 40).
- The best algorithm is BLSO because it provided the maximum accuracy value but the minimum execution time in all cases.
- The second best algorithm is RPO because it has a high flexibility to be implemented in real time applications, depended on a fewer parameters, and can provide solutions closer to the global optima.
- The third best algorithm is WSO because it depended on a fewer parameters and try to quickly prove optimal solution as possible.
- On the other hand, the worst algorithm is POA because it provided the minimum accuracy values and the maximum execution time values.

Thus, the proposed LSO is a simple structure that has a high flexibility to be implemented in real time applications for solving problems such as feature selection and also it depends on a small number of parameters. Additionally, LSO is an accurate algorithm that gives a global optimal solution with low convergence speed. Hence, LSO is a fast, accurate, and efficient algorithm but it needs to further improvement by hybridizing the LSO with other algorithms and it needs to be applied on many different applications to prove that it is applicable for different problems. The pros and cons of LSO algorithm are provided in Table 18.

9. Conclusions and future research

In this paper, a new natural inspired meta-heuristic optimization algorithm called Leopard Seal Optimization (LSO) algorithm was introduced as a swarm intelligence algorithm. The main idea of introducing LSO is to guarantee the

Table 18
The pros and cons of LSO algorithm.

Pros		Cons	
Feature	Description	Feature	Description
Speed	LSO is a fast algorithm because it can quickly select the best set of features.	Improvement	LSO needs to be improved by hybridizing it with other algorithms to be more effected algorithm.
Robustness	LSO is a robust algorithm that can deal with real time problems.	Variety	LSO needs to be implemented on many different problems to prove its applicability.
Precision	LSO can accurately reach to optimal solution.		
Effectiveness	LSO is an effective algorithm as it can give fast and accurate solution.		
Features number	BLSO can deal with dataset including large number of features and choose the accurate subset of them.		
Parameters number	LSO depends on a few number of parameters.		

exploration and exploitation of the search state for solving complex engineering problems and giving solutions closer to global optima. In fact, LSO proved its high efficiency compared to many recent swarm intelligence algorithms called Tuna Swarm Optimization (TSO), Pelican Optimization Algorithm (POA), Cat and Mouse-Based Optimization (CMBO), Aphid–Ant Mutualism (AAM), White Shark Optimizer (WSO), and Red Piranha Optimization (RPO). It is concluded that LSO outperformed other six algorithms based on t-test and Wilcoxon test as statistical analysis methods using five common benchmark functions and different numbers of search agents.

To test the proposed LSO on a real time problem, it was applied in a binary version called Binary LSO (BLSO) as a feature selection algorithm using medical dataset. In fact, BLSO proved its high performance compared to other algorithms because it provides the maximum accuracy and the minimum execution time values. Hence, BLSO outperformed other techniques using accuracy and execution time metrics with values reach to 97.62% of accuracy at the leopard seals number = 120 and 1314 of execution time at the leopard seals number = 40 when the iterations number = 400. Although LSO is a simple, fewer parameters, and flexible algorithm that can deal with real time problems, it needs to more improvements to give more accuracy values. Thus, we tend in our future research to provide a hybrid algorithm that combine LSO to other optimization algorithm for giving more accurate solutions. Additionally, LSO will be tested on many different applications to measure it applicability with different problems. LSO should be tested on many different datasets with different sizes.

CRediT authorship contribution statement

Asmaa H. Rabie: Study conception and design, Material preparation, Data collection, Analysis, Writing first draft of the manuscript. **Nehal A. Mansour:** Study conception and design, Material preparation, Data collection, Analysis, Writing first draft of the manuscript. **Ahmed I. Saleh:** Study conception and design, Material preparation, Data collection, Analysis, Writing first draft of the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We would like to thank the administration of the Faculty of engineering at Mansoura University and all the members of the faculty for creating a good work environment for the completion of this research in the faculty laboratories.

Funding

This research was not received any fund by any of the supporting institutions or companies.

Ethics approval

We (the authors) assure that we are committed to Ethics provided by the journal.,

Consent to participate

All authors agree to participate.

Consent to publish

All authors agree to publish in the journal.

References

[1] Singh R. Nature inspired based meta-heuristic techniques for global applications. Int J Comput Appl Inf Technol 2020;12(1):303–9.
[2] Sharma M, Kaur P. A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. Arch Comput Methods Eng 2021;28(5):1103–27, Springer.
[3] Monga P, Sharma M, Sharma A. A comprehensive meta-analysis of emerging swarm intelligent computing techniques and their research trend. J King Saud Univ Comput Inf Sci 2022;34:9622–43.
[4] Agrawal P, Abutarbo H, Ganesh V, et al. Meta-heuristic algorithms on feature selection: A survey of one decade of research (2009–2019). IEEE Access 2021;9:26766–91, IEEE.
[5] Gao Y, Zhou Y, Luo A. An efficient binary equilibrium optimizer algorithm for feature selection. IEEE Access 2020;8. 1409376-140963. IEEE.
[6] Sharma S, Singh G. Diagnosis of cardiac arrhythmia using swarm intelligence based metaheuristic techniques: A comparative analysis. EAI Endors Trans Perv Health Technol 2020;6(22):1–11.
[7] Wei B, Wang X, Xia X, et al. Novel self-adjusted particle swarm optimization algorithm for feature selection. Computing 2021;103:1569–97, Springer.
[8] George G, Raimond K. A survey on optimization algorithms for optimizing the numerical functions. Int J Comput Appl 2013;61(6):41–6.
[9] Hameed S, Hassan W, Latiff L, Muhammad F. A comparative study of nature-inspired metaheuristic algorithms using a three-phase hybrid approach for gene selection and classification in high-dimensional cancer datasets. Soft Comput 2021;25:8683–701, Springer.
[10] Mirjalili S, Mirjalili S, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61.
[11] Akyol S, Alatas B. Plant intelligence based metaheuristic optimization algorithms. Artif Intell Rev 2017;47:417–62, Springer.
[12] Alatas B, Bingol H. Comparative assessment of light-based intelligent search and optimization algorithms. Light Eng 2020;28(6):51–9.
[13] Alatas B, Bingol H. A physics-based novel approach for travelling tournament problem: Optics inspired optimization. Inf Technol Control 2019;48(3):374–88.
[14] Bingol H, Alatas B. Chaos based optics inspired optimization algorithms as global solution search approach. Chaos Solitons Fractals 2020;141:1–12, Elsevier.
[15] Sharma M, Kaur P. A comprehensive analysis of nature inspired meta heuristic techniques for feature selection problem. Arch Comput Methods Eng 2021;28(5):1103–27, Springer.
[16] Khoyetskyy P. Monitoring of the leopard seal population (Hydrurg leptonyx) in waters of the Argentine Islands (Ant-arctica). Theriol Ukr 2020;19:138–47.
[17] Rashid M. Tiki-taka algorithm: a novel metaheuristic inspired by football playing style. Eng Comput 2021;38(1):313–43.
[18] Guha R, Ghosh S, Ghosh K, Cuevas E, et al. Groundwater flow algorithm: A novel hydro-geology based optimization algorithm. IEEE Access 2022;10:132193-132211, IEEE.
[19] Rodríguez A, Camarena O, Cuevas E, Aranguren I, et al. Group-based synchronous-asynchronous grey wolf optimizer. Appl Math Model 2021;93:226–43, Elsevier.
[20] Krista V, Visse N, Rick B, Chris L, et al. Leopard seals (hydrurga leptonyx) in New Zealand waters predating on chondrichthyans. Front Mar Sci 2021;8:1–16.
[21] Agrawal P, Abutarboush H, Ganesh T, et al. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009–2019). IEEE Access 2021;9:26766–91, IEEE.
[22] Migallón H, Morenilla A, Rico H, et al. Multi-level parallel chaotic jaya optimization algorithms for solving constrained engineering design problems. J Super Comput 2021;77:12280–319, Springer.
[23] Alatas B. Uniform big bang–chaotic big crunch optimization. Commun Nonlinear Sci Numer Simul 2011;16(9):3696–703, Elsevier.
[24] Doraghinejad M, Nezamabadi-pour H. Black hole: A new operator for gravitational search algorithm. Int J Comput Intell Syst 2014;7(5):809–26.
[25] Aliman M, Ibrahim Z, Naim F, Nawawi S, et al. Performance evaluation of black hole algorithm, gravitational search algorithm and particle swarm optimization. Malays J Fund Appl Sci 2015;11(1):10–20.
[26] Deeb H, Sarangi A, Mishra D, et al. Improved black hole optimization algorithm for data clustering. J King Saud Univ Comput Inf Sci 2020;1–10. http://dx.doi.org/10.1016/j.jksuci.2020.12.013, Elsevier.
[27] Siddique N, Adeli H. Nature-inspired chemical reaction optimisation algorithms. Cogn Comput 2017;9:411–22, Springer.
[28] Altay E, Alatas B. Music based metaheuristic methods for constrained optimization. In: Proceedings in 2018 6th international symposium on digital forensic and security. ISDFS, Antalya, Turkey: IEEE; 2018, p. 1–6.
[29] Bouchekara H. Most valuable player algorithm: a novel optimization algorithm inspired from sport. Oper Res 2020;20:139–95, Springer.
[30] Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, et al. The arithmetic optimization algorithm. Comput Methods Appl Mech Engrg 2021;376:1–38, Elsevier.
[31] Monga P, Sharma M, Sharma S. A comprehensive meta-analysis of emerging swarm intelligent computing techniques and their research trend. J King Saud Univ Comput Inf Sci 2022;35(10):9622–43, Elsevier.
[32] Xie L, Han T, Zhou H, Zhang Z, et al. Tuna swarm optimization: A novel swarm-based metaheuristic algorithm for global optimization. Comput Intell Neurosci 2021;2021:1–22.
[33] Trojovský P, Dehghani M. Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. Sensors 2022;22(3):1–34.
[34] Dehghani M, Hubálovský S, Trojovský P. Cat and mouse based optimizer: A new nature-inspired optimization algorithm. Sensors 2021;21(15):1–30.
[35] Eslami N, Yazdani S, Mirzaei M, Hadavandi E. Aphid-ant mutualism: A novel nature-inspired meta-heuristic algorithm for solving optimization problems. Math Comput Simulation 2022;201(10):362–95.
[36] Braik M, Hammouri A, Atwan J, Al-Betar M, et al. White shark optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowl-Based Syst 2022;243:1–29, Elsevier.
[37] Zhiheng W, Jianhua L. Flamingo search algorithm: A new swarm intelligence optimization algorithm. IEEE Access 2021;9:88564–82, IEEE.
[38] Rabie A, Saleh A, Mansour N. Red piranha optimization (RPO): a natural inspired meta-heuristic algorithm for solving complex optimization problems. J Ambient Intell Humaniz Comput 2023;1–28. http://dx.doi.org/10.1007/s12652-023-04573-1, Springer.
[39] Hassan A, Abdullah S, Zamli K, Razali R. Whale optimization algorithm strategies for higher interaction strength T-way testing. Comput Mater Contin 2022;73(1):2058–77.
[40] Gao Z, Zhao J. An improved grey wolf optimization algorithm with variable weights. Comput Intell Neurosci 2019;2019:1–13, Hindawi.

[41] García S, Molina D, Lozano M, Herrera F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 2009;15:617–44, Springer.
[42] Rabie A, Ali S, Ali H, Saleh A. A fog based load forecasting strategy for smart grids using big electrical data. Cluster Comput 2019;22(1):241–70, Springer.
[43] Cabitza F, Campagner A, Ferrari D, Di Resta C, et al. Development, evaluation, and validation of machine learning models for COVID-19 detection based on routine blood tests. Clin Chem Lab Med (CCLM) 2020;59(2):421–31.
[44] Rabie A, Saleh A, Mansour N. A Covid-19's integrated herd immunity (CIHI) based on classifying people vulnerability. Comput Biol Med 2022;140:1–29, Elsevier.
[45] Kaggle. Diagnosis of COVID-19 and its clinical spectrum kaggle. 2021, https://www.kaggle.com/einsteindata4u/covid19 (Accessed 14 Jan 2021).