Pirates Swarm Search Algorithm: A Meta-Heuristic Search Algorithm for optimization problems

Milad Abolhasani Siahkali
Shahid Bahonar University Faculty of Engineering
Hamid Mirvaziri (h.vaziri@gmail.com)
Shahid Bahonar University Faculty of Engineering https://orcid.org/0000-0003-2863-3750
Abbas Bahrololoum
Shahid Bahonar University Faculty of Engineering

Research Article

Keywords: Pirates swarm search, non-bio-inspired, population-oriented, Meta-Heuristic

Posted Date: May 11th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1610967/v1

Pirates Swarm Search Algorithm: A Meta-Heuristic Search Algorithm for optimization problems

Milad Abolhasani Siahkali, Hamid Mirvaziri*, Abbas Bahrololoum

Computer Engineering Department, Faculty of Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.

Corresponding author: (hmirvaziri@uk.ac.ir)

Abstract: In this study, we have suggested a meta-heuristic algorithm based on swarm intelligence for single-objective optimization problems. Our inspiration comes from historical and behavioral fashions of pirates at 1600s and 1700s which also is known as the golden age of piracy. They navigate in the oceans to find treasures and we search the problem space to find a better solution. Triangle trade is the search space and each pirate ship is a search agent. A bunch of well-known benchmark functions are used to run some qualitative, quantitative and statistically tests to see the outcome of the suggested algorithm. Obtained results shows that the algorithm is able to explore and exploit in search space as expected and also has the ability of improving the average fitness of population and improving the best so far solution. At the end; results of suggested algorithm comparing to other well-known ones such as Particle Swarm Optimization, Gravity Search Algorithm, ~~and~~ Ranking Genetic Algorithm and Sparrow search algorithm shows that it is able to outperform ~~other~~ these algorithms in most of the studied cases.

Keywords: Pirates swarm search; non-bio-inspired; population-oriented; Meta-Heuristic

1. Introduction

Almost all applications in life is an optimization problem whether to minimize the cost and energy consumption, or to maximize the profit, output, performance and efficiency (Koziel & Yang, 2011). Algorithms for optimization can be roughly divided into two categories: exact algorithms and heuristics. Exact algorithms are designed in such a way that it is guaranteed that they will find the optimal solution in a finite amount of time. Heuristics do not have this guarantee (Sörensen, 2015). In solving real world optimization problems with a high dimensional search space, the exact search methods such as exhaustive search is not practical because the search space increases exponentially with the problem size (Rashedi & Nezamabadi pour, 2009). This means that the optimal solution cannot be found by the method of exhaustion in a limited time; it is thus important to develop an effective and efficient algorithm for this purpose. These methods are looking for a near optimal and an acceptable solution at an acceptable time using acceptable resource levels (Geem & Kim, 2001) Many meta-heuristic algorithms have been proposed to solve highly complicated optimization problems (Chou, et al., 2018). Examples of these problems are deterministic, stochastic, or population-based algorithms. These meta-heuristic algorithms can be classified into two main categories: single solutions like SA or multiple solutions like GA (Goldberg & Holland, 1988), PSO (Eberhart & Kennedy, 1995), HS (Geem & Kim, 2001), etc. Population-based algorithms with multiple solution are divided into two main categories of evolutionary and swarm and each individual factor in these categories behaves in an unsupervised and random behavior with respect to its neighboring space. Local laws lead to evolve swarm algorithms without any connection to global patterns and self-organized agents interactions. (Mishra, et al., 2013) Swarm intelligence algorithms including ant colony optimization (ACO),

Field Code Changed

particle swarm optimization (PSO), firefly (Xin-She Yang. & Xingshi He. 2009.), bee algorithm and gravitational search algorithm (GSA) and Sparrow Search algorithm (SSA-2019) are some of the most well-known algorithms. These algorithms are suitable for –optimization problems (Mavrovouniotis, et al., 2016) . In fact swarm intelligence techniques mimic the intelligence of swarms and creatures in nature. The main foundation of these algorithms originates from the collective behavior of a group of creatures (Mirjalili & et al, 2017). Swarm intelligence algorithms are divided based on imitation or sign oriented; biological or non-biological; memory-full or memory-less. Search space is all possible solution of the problem. They have some special properties such as being easy to learn and use. Being flexible to adapt and solve different kind of problems and derivation-free makes them useful while solving real world problems has expensive to solve with unknown derivation in some cases. Usually global optimum or local optimum can be found in a search space. Local Optimum is an optimal solution in just a part of search space and not necessarily the best solution to the problem. In contrast, global optimum is the best solution in the search space. Fitness function is designed according to the specifications and features of the problem. Objective of maximizing problems is to find the highest peak while in minimizing ones is to find the deepest valley. Swarm intelligence algorithms are stochastic, local optimum avoidance which is helpful when solving real world high dimensional complex problems. Search algorithms must have exploration and exploitation features to achieve a suitable solution. Exploration is the ability of moving freely in the search space regardless of previous results in order to find new locations. On the contrary, exploitation is the ability of finding better solutions in regards to the previous findings of the algorithm. Algorithms with the capability of exploration and customizable and flexible exploitation are the best option to solve problems with complex irregular conditions in a space which includes a large number of peak and Plateau (Reid, 2015). The aim of this article is designing of such an algorithm, according to the history of pirates in the golden age of piracy when well-known pirates like Jack Rackham, Blackbeard, Captain Kidd and Henry Every, are emerged as the best of them. Proposed algorithm work as follows:

- pirate ships are agenets looking for best fitness positions by moving in problem space as parts of the sea.
- To gain more wealth, they tend to identify and move towards greater wealthy ships.
- The amount of ship wealth is determined by the location of that ship.
- This amount of wealth and fitness has a direct impact on making decisions for speed control and ship movement in the next step.
- Wind interference and some creative changes make search agents as natural as possible for simulating ships movement.

The rest of this article to introduce such algorithm is organized as follows: Section 2 presents some previous works on the field. At section 3 we will introduce the main idea of suggested algorithm. Section 4 will be used to formulate the algorithm and at the section 5 we will perform a set of standard tests and experiments to verify the performances and quality of the suggested algorithm. At the end, section 6 will concludes the work and suggests some future works.

Formatted: Font: (Default) +Headings CS (Times New Roman), 12 pt, Font color: Text 1

Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Font: (Default) +Headings CS (Times New Roman), 12 pt

Formatted: Font: (Default) +Headings CS (Times New Roman), 12 pt, Font color: Text 1

2. Previous works

Imperialist competitive algorithm (Atashpaz-Gargari & Lucas, 2007) is a meta-heuristic search method that takes advantage of the colonialism phenomenon in history. The algorithm starts with a certain number of the population. Each individual population is considered a country and any country alone can be colonized and colonizer. A certain number of colonies are under colonial countries control due to their power.

The main question in Harmony Search is whether possible to provide a new algorithm with better solution and less number of repetitions than other existing algorithms. This new algorithm is the result of a music phenomenon called "Search for a better harmony". The general idea is based on the principle that harmony can be improved with practice, as well as optimal situation can be improved with repetition. (Geem & Kim, 2001)

PSO algorithm is a meta-heuristic, bio-inspired, population and imitation-based algorithm. Generally in different types of animals, including a variety of birds and fish, visible behavioral group is organized without a leader. PSO algorithms inspired from the behavior of birds and fish which combine personal and social experiences in order to find the optimal solution. (Eberhart & Kennedy, 1995)‹

GSA is inspired by the law of gravity and the mass concept. Searching agents in this algorithm are a set of objects. According to the laws of physics all materials attract each other by gravity and heavier objects have a greater impact than light objects. In this algorithm, the gravitational force is considered as an interface for data transfer and communicate between objects. Therefore heaviest object is the optimal point of interest in ideal conditions. Objects follow by gravity and moving law in GSA. In the gravity law, any object, absorb other objects by a force directly proportional to their masses and inversely proportional to the square of the distance between them. (Rashedi & Nezamabadi pour, 2009).

One of the latest algorithm published recently in this area is SSA as an effective optimization technique, which simulates the group wisdom foraging and anti-predation behaviors of sparrows. It can quickly and accurately search its required parameters and obtain optimal solution. However like other algorithms SSA also prone to fall into local optimum and population diversity decreases with the increase in iteration times (Xue & Shen 2020.)

Formatted: Font color: Auto

Formatted: Font color: Auto

Formatted: Font: (Default) Times New Roman, 12 pt

Formatted: Font color: Auto

Formatted: Font: (Default) Times New Roman, 12 pt, Font color: Auto, Pattern: Clear

Formatted: Font color: Red

3. Motivation and originality

Piracy is known as the act of robbery or similar criminal acts by a ship from other ships or coastal areas. People who do this are called pirates. Piracy is occurred since people use the sea for trade. The first period of trade triangle is formed when goods transferred from Europe to Africa. The second period of triangle trade is formed when slaves brought to America and the third period and final one formed during transportation of farm products from America to Europe. The third one refers to the tendency of merchant ships to travel between three continents instead of two ports in order to maximize the profits. Appropriateness of wind and water flow directions is the other reasons for using this route by mariners.

3.1. Search space and searching agents

Due to the comments mentioned above, the activities of pirates in triangle trade and the golden age of piracy was the motivation to design an algorithm. Imagination of issues like communications, wind and favorable currents motion and appearance of the most famous pirates, is used for ideation so this triangle formed a conception for problem space and pirates' ships are searching agents in this algorithm.

Life style and behavior of well-known pirates is the cornerstone of the idea and basis of operators design in proposed algorithm in conjunction with their estimated wealth and dominant strategy of attack. This wealth is evaluated by the location of the ship in the sea and proportional to the fitness of each ship. In each round of proposed algorithm, b top pirates are kept with highest fitness to

give them top awards and try to make more diversity for their movements by using their ships location in the search space.

3.2. Proposed Algorithm

Proposed method inspired by the above ideas allow agents search the problem space in lapse of time and try to find an optimal solution for the problem by defining the search space in discrete time with some legislation. Portions of triangle trade includes sea, islands and landmasses, are considered as the search space. Each point can be defined in the preset space in a multi-dimensional coordinate system. Searching agents are pirate ships that are scanning intelligently in order to increase their wealth and authority in the search space and have a clear position in the search space at time t. Each ship individually has some features such as location, authority, the sails and status information of the best ship location. All of these issues are determined due to the present location of ship and thetical laws.

Position of ship i is deliberated by equation (1) in a d dimensional space i.e. this position is x_i:

$$x_i = (x_i^1, x_i^2, \dots, x_i^d) \tag{1}$$

In the beginning position of all ships is defined randomly with a uniform distribution in the boundary of the problem according to equation (2):

$$x_i^d(t = 1) \sim U(x^{d,low}, x^{d,high}) \tag{2}$$

Every ship is located at a point and have a wealth according to its location. This wealth refers to the fitness (competency) of the ship and display it by $fit(x_i)$. After the initial translocation of ships and calculating their competency, it is necessary to record current position of each ship in its personal map (journal). In later iterations, only if the new ship's position has more fitness than its current position in its personal map then new values will be replaced. According to the equation (3) if m ships participate in search, the best position of each ship has been visited so far is available, and the best position that the $i'th$ ship has been ever visited is available by map_{p_i}

$$MAP_p = (map_{p_1}, map_{p_i}, \dots, map_{p_m}) \tag{3}$$

In the next step, a ship with the highest fitness is chosen as the leader (captain Every) according to the equation (4). In the early stages, none of the participating ships in the searching process will not see the leader and the position of the leader have a little impact on the movement of other ships. This means more problem space searching is possible by other ships in the early stages of the algorithm.

$$x_{leaedr} = \text{argmax}\big(fit(X)\big) \tag{4}$$

When the leader is identified and personal maps (journals) are recorded, its turn to generate and record treasure maps. Treasure maps are the top n point except leader's location which is seen by all the population up to this moment. Number of these maps is determined parametrically by the user. map_{t_1} refers to the first treasure map and has the highest fitness after leader's location. The second map has definitely a lower fitness than the first map as seen in equation (5).

$$MAP_t = (map_{t_1}, map_{t_2}, \dots, map_{t_n}) \tag{5}$$

We need to identify top b pirate with highest fitness, which are displayed by TP in equation (6), and have a higher wealth (and are award nominated in our assumptions) in search space. Like the previous, the leader will not be somewhere in this list. b is specified by the user parametrically.

$$TP = (tp_1, tp_2, tp_3, \dots, tp_b) \tag{6}$$

$tp_1$is the best pirate after the leader and tp_2 to tp_b will be in the next positions. Now it is needed to figure out the distance of each ship with the point in which the leader is present to determine who is able to see the leader. In order to calculate this distance for each ship, Euclidian Distance is used in accordance with equation (7). $Vt_i(t)$ is the Euclidian distance of $i'th$ ship related to the leader's location at time t.

$$Vt_i(t) = \| x_i - x_{leader} \| \tag{7}$$

After determining distance between ships and the leader, each ship should adjust its own sail by equation (8) to get proportionate speed for exploration and exploitation with windflaw rate at time t.

$$sailsMode_i(t) = \begin{cases} Reefed\ sails, & Vt_i \le r \\ Full\ sails, & Vt_i > r \end{cases} \tag{8}$$

r is a radius where the leader is visible by other ships. Its value is determined parametrically as $r_{max}$by any user. This parameter increases linearly during the operation of the algorithm, in the form of equation (9) i.e. it will be increased from a very small value to reach r_{max}eventually.

$$r(t) = \frac{t}{T} \times r_{max} \tag{9}$$

In this way, after some time, there will be enough information about leader location and all pirate ships inform about its location gradually. Consequently, visibility does not necessarily mean direct observation of the leader. Therefore all ships are informed about leader's location in the final stages even from a long distance (unless r_{max} is determined wrongly or on purpose which does not allow this for some ships).

Traditionally sails was designed in square or triangular shapes. These shapes have their own features with particular advantages. The ships with square-shaped sail show their best performance when they moved in to the favorable direction of the wind while ships with a triangular got it when moved along the wind i.e. wind blow completely perpendicular to their sails (Sid meier's pirates, 2004). It is assumed that the ships have a triangular sail in the proposed algorithm and the direction of the wind direction is such that ships have their best performance. After being decided for sail to be set to a fully or Reefed shape, values proportional to these situation are adjusted as the sail angle related to the wind, for each ship according to the equation (10). This value is set zero for the leader to stay motionless in the best position.

$$Sails_i(t) = \begin{cases} \frac{\pi}{2}, & Full\ sails \\ \frac{\pi}{4}, & Reefed\ sails \\ 0, & Leader \end{cases} \tag{10}$$

Wind impact on ship $i'th$ movement can be achieved simplistically by equation (11) regardless of water and wind friction and wave and drag forces on the hulk with triangular-shape sail. In this equation w is wind speed, and φ is equal to cross section of sail area.

$$\delta_i = w(t)^2 \times \sin(Sails_i(t)) \times \varphi \quad (11)$$
$$\varphi = 1$$

The value of φ is set to 1 for all ships. Speed of each ship is controlled by the sail angle related to wind direction. In each repetition w is chosen randomly according to the equation (14) with a uniform distribution between w_{min} andw_{max}. Also, w_{min} andw_{max} values are calculated according to the equation (12) and (13) in the beginning.

$$w_{max}(t) = w_{start} - \left(\frac{t}{T} \times w_{start}\right) \quad (12)$$

$$w_{min}(t) = \begin{cases} w_{max} - w_{range}, & w_{max} - w_{range} \geq 0 \\ 0, & otherwise \end{cases} \quad (13)$$

$$w(t) \sim U(w_{min}(t), w_{max}(t)) \quad (14)$$

Larger numbers for w_{range} can have heterogeneous steps and a lot of distance in different iterations, while a smaller number has regular and close steps. Values greater than one for w_{start} lead to increased speed, dispersion, and consequently more exploration in the problem space. In case of stagnancy, high amounts of wind can be used to increase exploration. In this article, values above two are used to explore the problem space in the early stages, especially in the above dimensions. In nature, there may be moments of high velocity of the wind which suddenly decreases. In the proposed algorithm sudden reduction of the wind help convergence and more exploitation after reaching a certain amount of speed.

In the following, due to the fitness of different ships, the amount of δ is shared. Equation (16) is used for this purpose and the shared value is called η. m_i is equal to the mass of $i'th$ ship in this regard. In order to calculate the mass of $i'th$ ship, the amount of their fitness is mapped to their mass according to the equation (15) (Rashedi & Nezamabadi pour, 2009). worst(t) and best(t) are the worst and best population fitness at time t.

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} + 1 \quad (15)$$

Therefore η become half of δ for a ship with maximum fitness and shared for the rest of them related to their mass i.e. the ship with the more fitness has more weight and smaller movement and

meanwhile the ship with the lowest fitness has less weight and bigger movement proportionally giving by the equation (16).

$$\eta_i(t) = \frac{\delta_i}{m_i} \tag{16}$$

Move tendency of each ship is computed according to the situation, by equation (17_A, B or C). The status of each ship is determined according to Table 1.

$$Md_i(t) = \begin{cases} c_l \times r_1 \times (x_{leaedr} - x_i) + c_p \times r_2 \times \left(map_{p_i} - x_i\right), & A \\ c_m \times r_1 \times \left(map_{t_r} - x_i\right) + c_p \times r_2 \times \left(map_{p_i} - x_i\right), & B \\ c_t \times r_1 \times (tp_r - x_i) + c_p \times r_2 \times \left(map_{p_i} - x_i\right), & C \end{cases} \tag{17}$$

c_l, c_m, c_t and c_p are constants to weigh the ship movement tendency towards the leader, treasure map, top ships and personal map. $r_1$and r_2 are two random values between 0 and 1 to include more probability in movement; it also makes the assumption of knowing the location of the corresponding ship involved in the movement. $map_{t_r}$is a treasure map selected randomly from N available treasure map by Roulette wheel method and finally tp_r is the superior pirate chosen between top b pirates previously selected randomly by Roulette wheel method.

Table 1. Move tendency of each ship

Movement formula	Condition
A	When the leader is seen
B	When includes in best b pirate
C	otherwise

After calculating η and Md for each ship, the speed of $i'th$ ship at time $t+1$ is obtained by equation (18). v_i is assigned randomly at time $t = 0$.

$$v_i(t+1) = v_i(t) \times \eta_i(t) + Md_i(t) \tag{18}$$

Finally, position varying of a ship takes place according to equation (19).

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{19}$$

In addition to the above computation carried out in each iteration and in order to move ships, three legend, battle and storms operators are used to improve the algorithm performance. In continue they are explained. Legend operator tries to discover a new point that Captain Kidd's treasure

might be hided by available maps of best location of the most visited points. In other words, a new point is created and its fitness is evaluated by considering fragments of the existing treasure maps and putting these pieces together in a random way. If the new discovered point is better than the points in the maps, then the worst map will be replaced. In certain situations where the new point even has a better fitness than the leader, leader's position is changed and treasure maps are leave without any intact. In a three-dimensional problem where three maps exist like in Table 2, generated legend from these maps may have a better fitness demonstrated in Table 3.

Table 2. Treasure maps in a 3-D problem

x_1	x_2	x_3
1	2	3
4	5	6
1	8	9

Table 3.Generated legend from treasure maps

x_1	x_2	x_3
1	8	6

It's up to you how many legends you want to create in each iteration of algorithm. There can be one or more legends in each iteration of the algorithm, according to the needs and then their fitness are evaluated.

Next stage is called the battle. In this stage a number of top ships will be involved to find a better location. Number of battles in each iteration and number of ships involved in each battle is arbitrary. It is suppose that b battle will be occurred and in each battle two ships enter to a sea war with each other. The result of this battle is evaluating of a new point called ω in the problem space. This point is obtained according to the equations (20) to (22). In each battle, two ships are chosen to simulate the incident by Roulette-wheel from top b pirates.

$$\alpha = \frac{fit(x_1)}{fit(x_1) + fit(x_2)} \tag{20}$$

$$\beta = \frac{fit(x_2)}{fit(x_1) + fit(x_2)} \tag{21}$$

$$\omega = \alpha \times x_1 + \beta \times x_2 \tag{22}$$

According to the fitness level of ω, this operator can lead to capture or sink of a ship by an attacker in the early stages of the algorithm. If $fit(\omega)$ is greater than the fitness of both ships involved in the battle then battle result is capturing and the winner is the ship with higher fitness. Therefore, the loser will be considered as the ship's assets of the winner so the winner change its location to position ω and the loser is removed from the algorithm's memory. A new ship is produced in a

randomly position in the permissible space in order to preserve the initial population of the algorithms. Therefore loser pirate will find the opportunity to re-enter the sea with a new ship and begin to search again. In this case the captain's personal map (journal) of removed ship is cleared and the new ship would still have access to the best position where his captain has met. A capturing sample is shown in Fig 1. F represents the rate of ship's fitness before and after the battle.

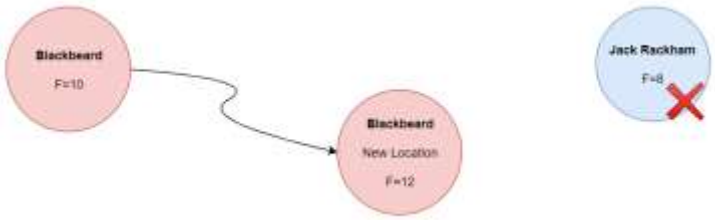


Figure 1. Battle ending of to a capture

Another possible status happens when $fit(\omega)$'s of one ships involved in battle is lower than the other. This situation will leads to sink the lower fitness ship by winner of the battle. For example, if fit (ω) of that ship equals to any values smaller than 10, it will be a battle loser, so is removed from the memory like before and the battle winner keep its former position as seen in Figure 2.

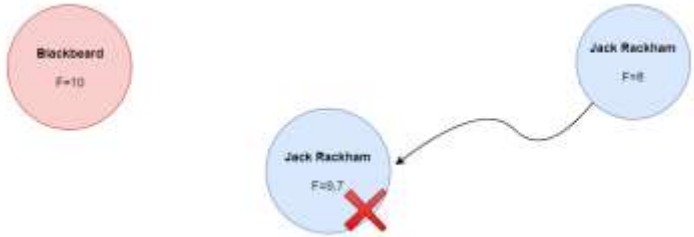


Figure 2. Battle ending up to a sink

As removing a ship from the memory and placing it in a new random position may cause to decrease the ability of the convergence so the occurrence of this action must be controlled in such a way that not affect the convergence at the end stages of the algorithm. For this purpose, only if the conditions of equation (23) is confirmed, a ship can be removed. λ is determined by the user. Results and intuitive tests show that 3 is a good value for λ.

$$t < \frac{T}{\lambda} \tag{23}$$

There is a storm operator for all ships in each iteration with P_h probability determined by the user according to the equation (24). This operator change ship's position in a completely randomized manner. In addition to the amount of P_h, the time in which algorithm is working has a straight

effect on the storm incident. The possibility of a storm incident should be controlled in such a way that rarely happened in the final stages of the algorithm, otherwise exploitation power and convergence of the algorithm is faced with problem. It is suggested that the operation of the storm is controlled by equation (23) and it operates only if this equation is feasible.

$$x_i^d(t+1) \sim U(x^{d,low}, x^{d,high}), \quad \begin{matrix} if\ r < P_h\ , \\ if\ i\ != leader \end{matrix} \tag{24}$$

r equals to a random value between zero and one. We intend to send some ships to explore other areas by applying the storm operator on the population. This operation give it a chance to get out of local optimums in certain situation. In continue all stages of proposed algorithm is described according to the specified ideas.

3.3. Pseudo-code of the algorithm

In the following, the general stages of the proposed algorithm are specified in details.

1. Setting and adjusting parameters
2. Basic Ship Positioning
3. Fitness computation of all ships
4. Updating ships Journals
5. Leader Identification
6. Treasure Maps Updating
7. r updating; leader's visibility status
8. Adjusting angle of sail for all ships
9. Wind Rate updating
10. Updating list of top ships
11. Updating the speed and position of all ships
12. Legends producing and evaluating
13. Battle
14. Storm
15. If the termination condition fails, go to the third step

In the next section, an example is solved according to specified order of the algorithm.

~~3.3.1.~~3.4. Solving an example

In this section, an optimization problem is solved in two dimensions by the proposed algorithm. Concerning function is shown in equation (25). The aim is minimizing this function in a continuous space in the range of [-100, 100]. It is in fact the sphere evaluation function which is displayed in Figure 3.

$$F(X) = \sum_{i=1}^{n} x_i^2 \tag{25}$$

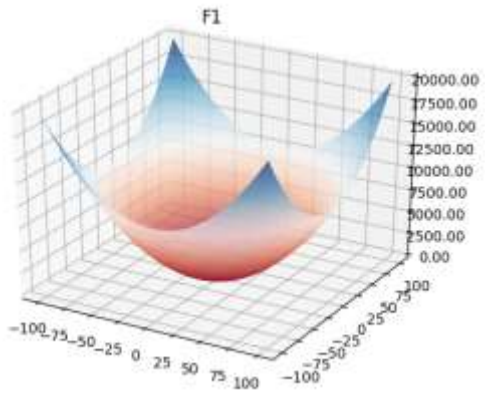


Figure 3. Sphere benchmark function

The number of search ships is assigned to 30, and 10 of these ships are ranked as top ships in each iteration, and are used as top 10 pirates. Termination condition is considered the number of the algorithm iterations. A very small value is considered for the occurrence of storm. The other parameters are assigned according to Table 4.

Table 4. Parameters initialization to solve sample problem

Values		**Parameters**	
1		r_{max}	
0.02	1	w_{range}	w_{start}
3		λ	
30		numbers of all ship	
10		numbers of best b pirates	
3		numbers of treasure maps	
0.001		(P_h) Storm incident probability	
$c_l = 2$	$c_t = 1.3$	$c_m = 1.5$	$c_p = 2.1$
100		$max\ iter$	

After parameters setting and accidental assignment of search agents, the location of all ships is displayed in Figure 4 in the first iteration of the algorithm after identifying the leader, the top pirates and determination of treasure maps. As it is shown in this Figure, the ships are distributed in problems space between [-100,100].

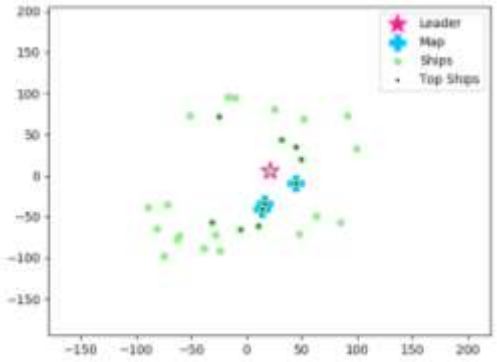


Figure 4. Positions of ships at iteration 1

Figure 5 demonstrate the leader's position, treasure maps and the pirates' ships in $5'th$ iteration. It can be seen in this Figure that ships are moving towards the best position identified up to this moment.

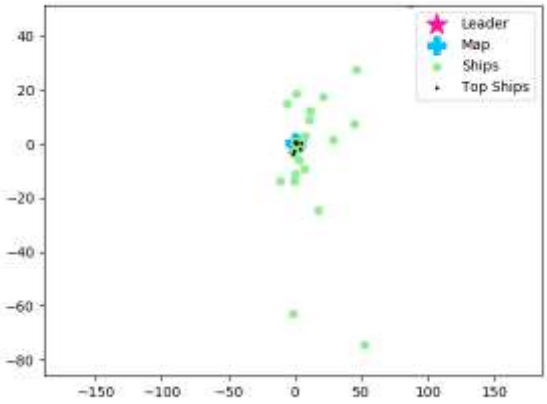


Figure 5. Positions of ships at iteration 5

Figure 6 shows $50'th$ iteration of the algorithm where almost all ships are around [0, 0] looking for optimum point. This figure is scaled enough to make the ships separable and viewable.

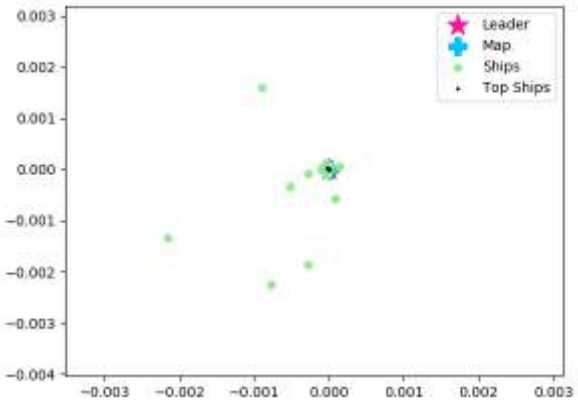


Figure 6. Positions of ships at iteration 50

Moving to position [0, 0] which is actually the optimal solution verifies the performance operation of the algorithm. Ships position in the last stage (100'th iteration) is illustrated in Figure 7.

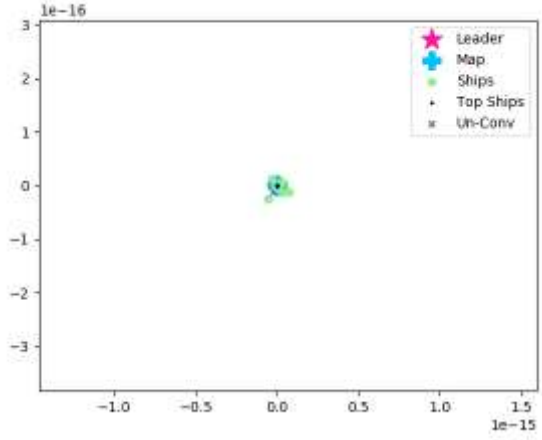


Figure 7. Positions of ships at iteration 100

One of the best ways to evaluate the performance of an algorithm is evaluation of the best so far solution and the average of its population fitness over time which is demonstrated in Figure 8.

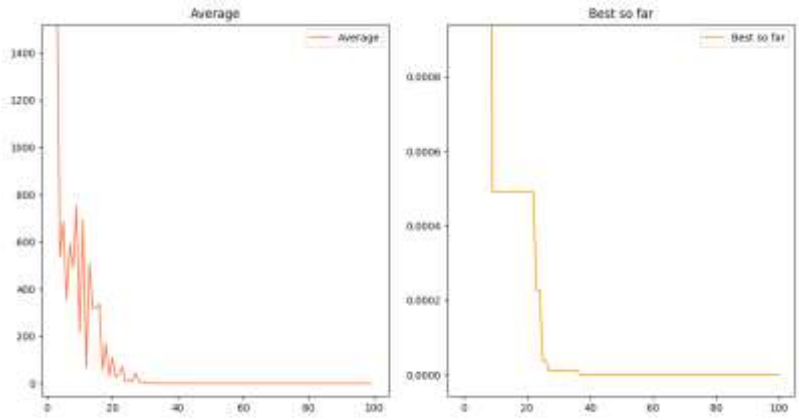


Figure 8. Best so far and population's average fitness during algorithm run time

Last two figures display natural function of the algorithm in solving the sample problem so that population besides the best-so-far solution are improved and converged at the end. During the operation of the algorithm which included 100 repetitions, the best so far solution is changed 66 times and final result was $1.5360195046038782 \times 10^{-35}$ after 100 iteration. To ensure this result is not achieved accidentally and in order to examine the algorithm more precisely, the evaluation function resolved 30 times by the algorithm and the results are given in Table 5. Issues such as mean and standard deviation reveal the correctness of the algorithm and verify the previous results.

Table 5. Obtaine results from 30 independent run

$1.0279520194132663 \times 10^{-32}$	Mean
$3.7124433104148216 \times 10^{-32}$	Standard deviation
$2.2152900664419454 \times 10^{-40}$	best so far solution
$1.935614089901295 \times 10^{-31}$	worse so far solution

The proposed meta-heuristic, non-biological, population-oriented, and swarm-based algorithm is memory-full and inspired from Pirate's life on the Golden Age of piracy to search problem space. One of the key features of this algorithm is its high motion variation of search agents that can be changed and improved in different forms. Another feature is its remarkable number of customizable parameters. These parameters give the ability of adapting its behavior for wide range and type of problems to be searched and resolved. Although it may be assumed that user should have sufficient information on the problem in order to set the parameters, but these values are conjecturable in general for setting a significant number of problems and there is no need for accurate knowledge and information about the problem. However, presence of these parameters provide enough power for mindful user to set the algorithm in such a way that makes much better solutions.

4. Evaluation

One of the most common tools to evaluate the quality of solutions in solving the single objective problems is the convergence curve chart. It is better to evaluate the capabilities of the algorithm by standard benchmark functions. These functions are designed to be able to evaluate the algorithm in a standard way and enable to compare it easily and confidently with other algorithms. Each group of these functions has a special characteristics. Uni-modal functions give a decent criteria to evaluate convergence speed and exploitation of the algorithm (Mirjalili & et al, 2017). In contrast, multi-modal functions have a global optimum besides a large number of local optimum that are increased when dimension increased. Therefore these functions are a good option to investigate the ability of the algorithm's avoidance to fall in local optimum and its exploration strength (Mirjalili, 2014).

4.1. Quality Evaluation

Different qualitative scales are displayed for qualitative evaluation of the algorithm, in this section. To produce these results, parameters are set according to Table 6 and the average of best so far is shown in Table 7. Performance evaluation of proposed algorithm to find an optimal solution is compared with RGA (Ranking Genetic Algorithm), PSO and GA algorithm in Figure ~~Table~~ 8 to Figure ~~Table~~ 13.

The first kept quality scale is the ships' path during the operation of the algorithm. Since, presenting the path of all search agents in all aspects of the problem and analysis of their result is a difficult hence the first ship's path is stored and displayed in the first dimension. This data shows performing of exploration and exploitation of proposed algorithm during problem solving. Evaluation of this graph in different functions indicates that the ship has irregular movement at the beginning and relatively uniform movement at the final stages. In fact, it reveals that ships initially attempted to explore in the problem space and eventually redirected to the global optimum by exploitation to find a better solution.

Table 6. Parameters used to initialize algorithm for qualitative evaluation

30		number of ships	
5		numbers of b best pirates	
2		number of treasure maps	
0.002	1	w_{range}	w_{start}
1	3	r_{max}	λ
0.001		Probability of storm	
2		Benchmark functions dimensions	
$c_l = 2$	$c_t = 1.2$	$c_m = 1.2$	$c_p = 1.4$

500	Iterations at each independent run
30	Number of independent run

Table 7. Best so far mean during qualitative evaluation

1.34E-32	**F13**	8.14E-185	**F1**
0.998	**F14**	4.5E-60	**F2**
0.0003	**F15**	5.8E-239	**F3**
-1.031	**F16**	2.2E-54	**F4**
0.397	**F17**	0	**F5**
2.999	**F18**	0	**F6**
-3.862	**F19**	0	**F7**
-3.266	**F20**	-816	**F8**
-5.599	**F21**	0	**F9**
-5.568	**F22**	7.99E-16	**F10**
-6.238	**F23**	0	**F11**
		4.66E-23	**F12**

The ships' paths show the sequence of attempting to explore and exploit in the problem space. However, this data is not able to show whether these operations are efficient or not i.e. the algorithm is able to improve randomly generated population in the initial stage and find an approximate solution or not. Thus, the average amount of fitness of all the participating ships, along with the best so far solution over time, was stored along with the ship's path in Figure ~~Table~~ 14 to Figure ~~Table~~ 19. Graphs of average fitness of population shows a descending trend. Because all the benchmark functions are minimization functions so the results prove that the algorithm has the ability to improve the population fitness during the operation process. Another notable pattern is the uptrend of the population fitness, especially at the early stages which exhibit the exploration (Mirjalili & et al, 2017)

~~Table~~ Figure 8 – Comparing algorithm with GA, GSA, PSO – (F1, 2 Dimension)

500 Iteration – 30 independent run

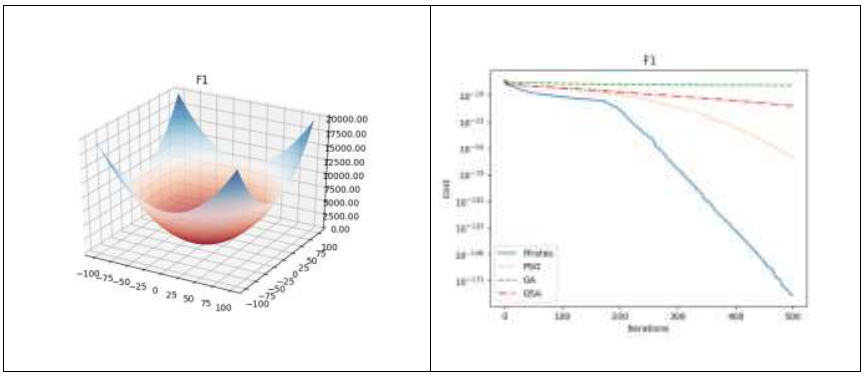


Figure ~~Table~~ 9 – Comparing algorithm with GA, GSA, PSO – (F5, 2 Dimension)
500 Iteration – 30 independent run

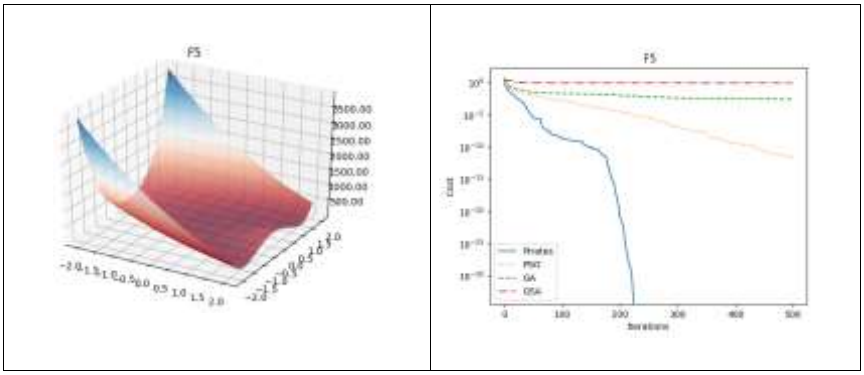


Figure ~~Table~~ 10 – Comparing algorithm with GA, GSA, PSO – (F9, 2 Dimension)

500 Iteration – 30 independent run

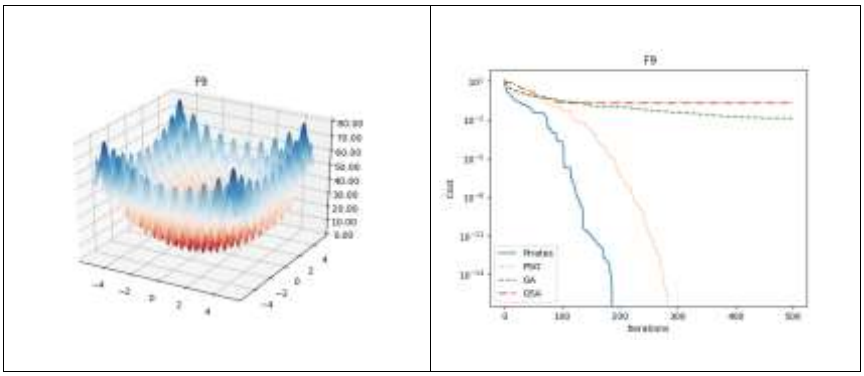


Figure ~~Table~~ 11 – Comparing algorithm with GA, GSA, PSO – (F10, 2 Dimension)
500 Iteration – 30 independent run

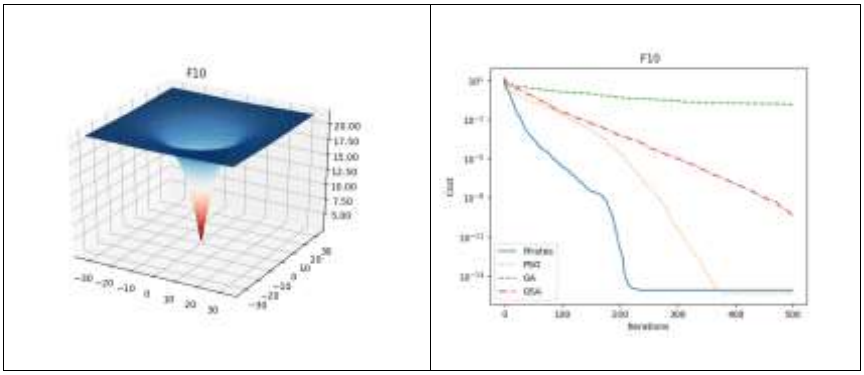


Figure ~~Table~~ 12 – Comparing algorithm with GA, GSA, PSO – (F11, 2 Dimension)

500 Iteration – 30 independent run

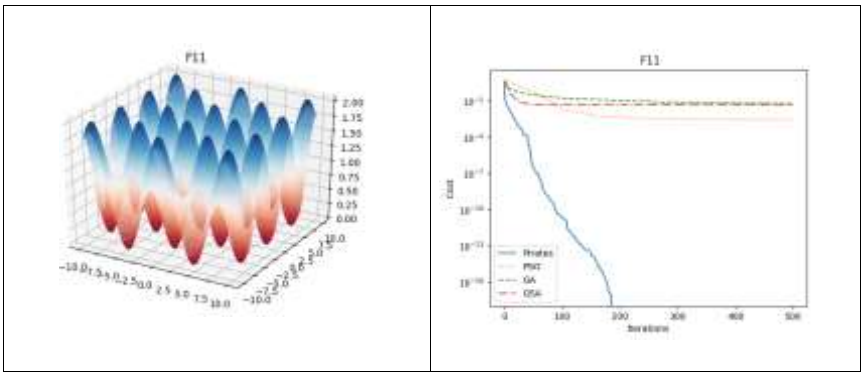


Figure ~~Table~~ 13 – Comparing algorithm with GA, GSA, PSO – (F18, 2 Dimension)
500 Iteration – 30 independent run

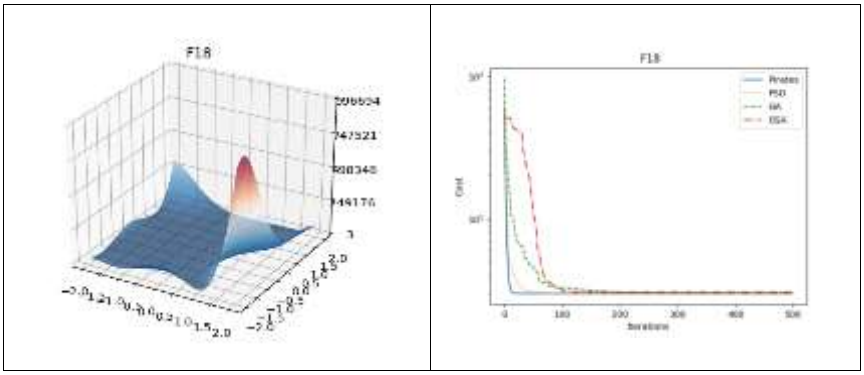
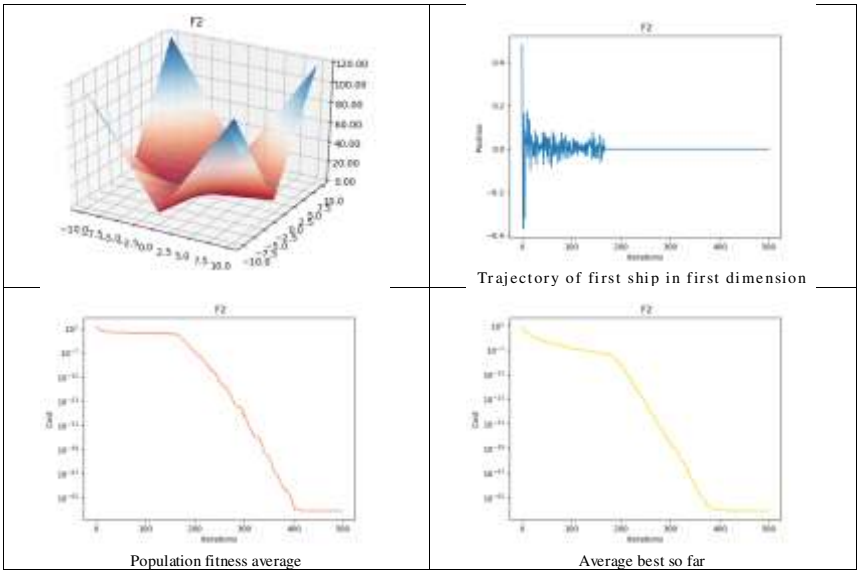


Figure ~~Table~~ 14 – Qualitative results of algorithm – (F2, 2 Dimension) – 30 Search Agent - 500 Iteration 30 independent run

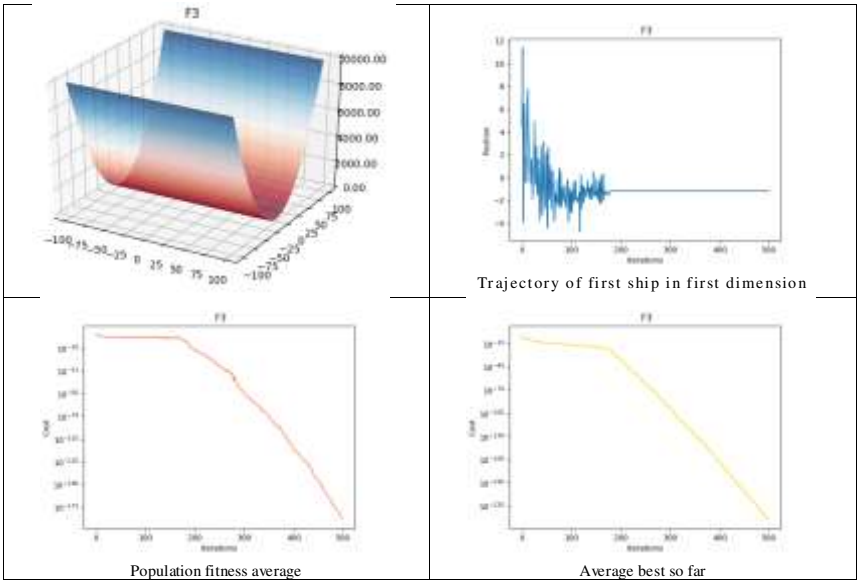


Trajectory of first ship in first dimension

Population fitness average

Average best so far

Figure ~~Table~~ 15 – Qualitative results of algorithm – (F3, 2 Dimension) – 30 Search Agent - 500 Iteration
30 independent run

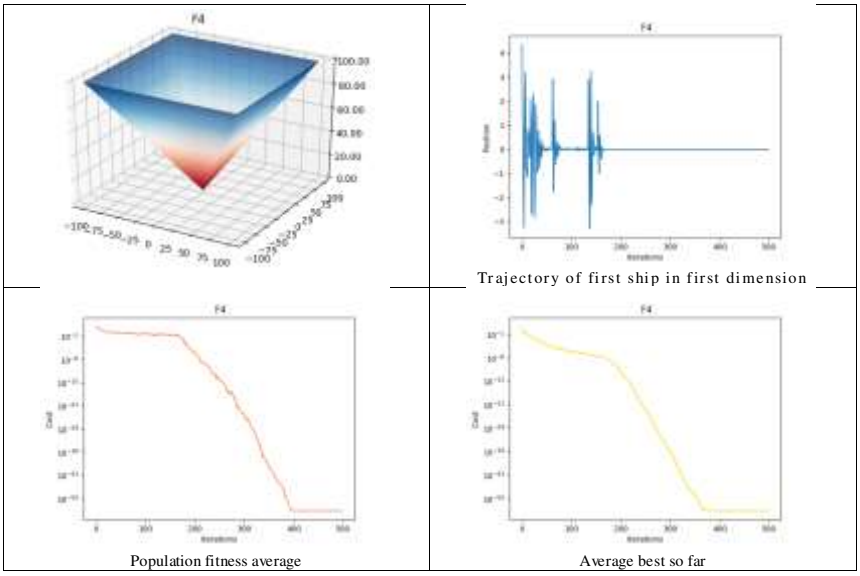


Trajectory of first ship in first dimension

Population fitness average

Average best so far

Figure ~~Table~~ 16 – Qualitative results of algorithm – (F4, 2 Dimension) – 30 Search Agent - 500 Iteration 30 independent run

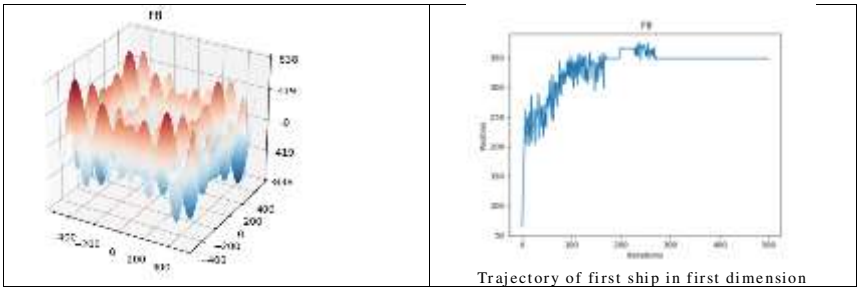


Trajectory of first ship in first dimension

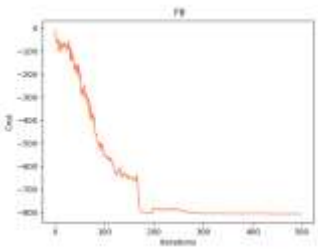
Population fitness average

Average best so far

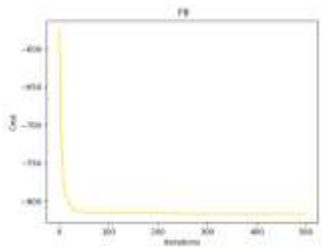
Figure ~~Table~~ 17 – Qualitative results of algorithm – (F8, 2 Dimension) – 30 Search Agent - 500 Iteration 30 independent run



Trajectory of first ship in first dimension

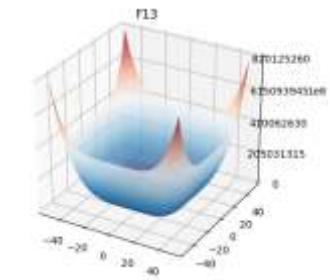


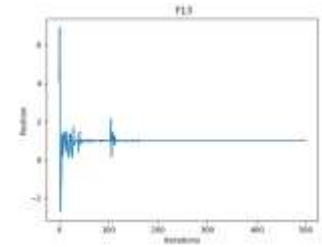
Population fitness average



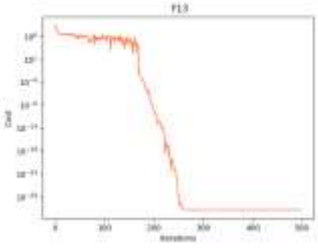
Average best so far

Figure ~~Table~~ 18 – Qualitative results of algorithm – (F13, 2 Dimension) – 30 Search Agent - 500 Iteration 30 independent run

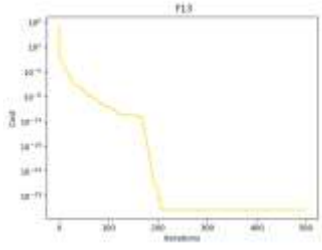




Trajectory of first ship in first dimension

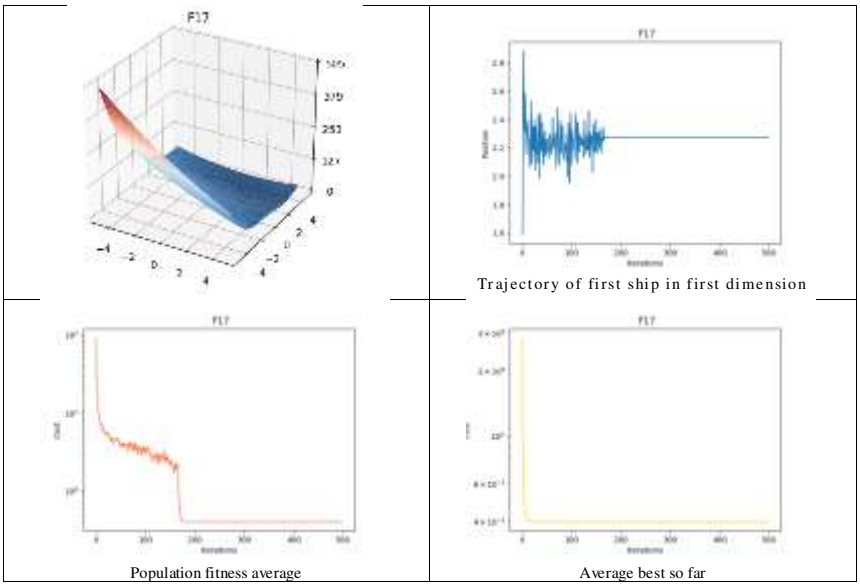


Population fitness average



Average best so far

Figure ~~Table~~ 19 – Qualitative results of algorithm – (F17, 2 Dimension) – 30 Search Agent - 500 Iteration 30 independent run



Trajectory of first ship in first dimension

Population fitness average

Average best so far

Although the population fitness rate demonstrate that population status is improving over the time of operation, it will not be able to reveal the improvement of the estimated solution status regarding to the global optimum. For this purpose, the best so far solution over time is stored and displayed. After evaluation of these graphs, fitness of estimated global optimum is improving during the operation of the algorithm, however this improvement is not the same and stable in all problems. Obtained results show that the algorithm has a big ability to improve estimated solution of the global optimum in uni-modal functions, representing the great exploitation capability. The results of this qualitative assessment show the algorithm explore and exploit the problem space during its operation and has the ability to improve the status of the random population generated in the first stage. It is also able to improve its estimated solution of the global optimum and exploit the important locations of the problem space at the searching time.

4.2. Quantitative ~~Quantity~~ Evaluation

Although it can be concluded that the algorithm shows a correct action by considering the above qualitative criteria but it cannot be deduced how good the algorithm is working. Numerical scales are used to evaluate this matter. Two Tests are performed to make this evaluation. In each test, the proposed algorithm is run independently 30 times on each benchmark function. Mean and standard deviation are obtained in the first Test, and mean and median are obtained in the second test as shown in ~~Table~~ Figure 20 to ~~Table~~ Figure 24. In order to verify obtained results and evaluate its

performance, PSO, GSA and GA are chosen to be compared. The number of dimensions of benchmark functions in the first test is equal to 10 with 30 search agents and maximum iteration of 500 per each independent run of the algorithm. In the second Test, the dimensions of benchmark functions is considered 30 with 50 search agents and the maximum iteration of 1000.

In order to keep fairness and considering that the results of any algorithm are dependent to initialization and interval of the parameters, results of the proposed algorithm is compared with results of Dragonfly algorithm's article (Mirjalili, 2015), in same conditions for first test. In the second test, obtained results are compared again with results introduced in GSA (Rashedi & Nezamabadi pour, 2009) in the same conditions to ensure providing necessary settings and achieving best performance on the desired functions.

Table ~~20~~8. Evaluation on uni-modal benchmark functions in 10 dimension, 30 search agents, 500 iteration and 30 independent run

RGA		PSO		SSA		Proposed Algorithm		Function
Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	
324.9262	748.5972	1.31E-17	4.2E-18	1.84	4.91E-110	1.68E-39	3.12E-40	F1
1.533102	5.971358	0.009811	0.003154	1.03	1.98E-65	1.98E-10	4.95E-11	F2
994.2733	1949.003	0.003311	0.001891	2.5	4.64E-63	4.46E-14	8.29E-15	F3
2.605406	21.16304	0.002515	0.001748	1.15	2.139E-46	1.69E-17	3.65E-18	F4
85,007.62	133307.1	80.12726	63.45331	4.73	9.41E-06	0.79522	6.67921	F5
229.6997	563.8889	1.38E-16	4.36E-17	4.97	2.18E-08	1.21E-05	3.84E-06	F6
0.072571	0.166872	0.003583	0.005973	0.0	1.2E-220	1.47E-74	2.86E-75	F7

Obtained results on uni-modal functions show the proposed algorithm is working better than the others in almost all cases. The lower average value indicates proposed algorithm has better performance compared to PSO and lower standard deviation values prove this advantage has a high stability. It is only loser in F6 function. As stated before, uni-modal functions are decent to check the power of exploit these results ensure proposed algorithm has a proper exploit power.

Table ~~21~~9. Evaluation on multi-modal benchmark functions in 10 dimension, 30 search agents, 500 iteration and 30 independent run

RGA		PSO		SSA		Proposed Algorithm		Function
Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	

Formatted: Tab stops: 1.11 cm, Left + 1.55 cm, Centered
Formatted Table
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Tab stops: 1.01 cm, Left + 1.52 cm, Centered
Formatted Table

164.4776	-3407.25	1152.81	-4841.29	629.54	-3440.71	282	-3353.25	F8
6.66936	25.51886	7.879807	10.44724	0	0	1.13E-12	2.21E-13	F9
1.271393	9.498785	0.601817	0.280137	0	4.44E-16	2.19E-13	4.11E-14	F10
3.62607	7.719959	0.035067	0.083463	0	0	0	0	F11
5820.215	1858.502	2.71E-10	8.57E-11	2.47E-08	6.81E-09	8.45E-5	3.76E-5	F12
87,736.76	68,047.23	0.004633	0.002197	2.62E-07	8.42E-08	0.002741	0.000733	F13

Evaluating the results on multi-modal functions show that proposed method is working better in all of them except F8 and F13 functions. In contrast to the uni-mode functions, they are a good option to check the algorithm's exploring power. Since multi-modal functions have a large number of local optimum, only algorithms with a proper exploring power have the potential of achieving the global optimum. The above results show the ability of the algorithm to explore problem space efficiently. Finally, the mean values and standard deviation indicate how much the algorithm has acted powerful in exploration and escaping of local optimum. In order to investigate the ability of the algorithm to solve complex problems with higher input number, the second test was performed on more benchmark functions in 30 dimensions and the results were compared with PSO, GSA and original version of GA shown in Table ~~22~~10, ~~23~~11 and ~~24~~12.

Table ~~22~~10. Evaluation on uni-modal benchmark functions in 30 dimension, 50 search agents, 30 independent run

RGA		GSA		PSO		SSA		Proposed Algorithm		Function
Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	
21.87	23.13	7.1E-11	7.3E-11	1.2E-3	1.8E-3	1.34E-316	3.44E-296	3.5E-47	5.0E-25	F1
1.13	1.07	4.07E-5	4.03E-5	1.9E-3	2.0	1.51E-167	1.48E-163	2.2E-18	2.8E-8	F2
5.6E+3	5.6E+3	0.15E+3	0.16E+3	2.2E+3	4.1E+3	3.62E-177	3.28E-146	8.5E-43	5.7E-26	F3
11.94	11.78	3.7E-6	3.7E-6	7.4	8.1	1.97E-160	1.02E-138	2.7E-34	9.5E-32	F4
1.0E+3	1.1E+3	25.18	25.16	1.7E+10	3.6E+4	7.64E-07	3.71E-06	28.43	28.30	F5
24.55	24.01	7.7E-11	8.3E-11	6.6E-3	1.0E-3	1.63E-09	1.45E-08	1.0E-3	1.0E-3	F6
0.06	0.06	0.015	0.018	0.04	0.04	0	0	1.0E-87	8.6E-16	F7

Table ~~23~~11. Evaluation on multi-modal benchmark functions in 30 dimension, 50 search agents, 1000

Formatted: Font: (Default) +Headings (Calibri Light), 10 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Headings (Calibri Light), 10 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Headings (Calibri Light), 10 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Headings (Calibri Light), 10 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Headings (Calibri Light), 10 pt, Font color: Text 1, English (United States)
Formatted Table
Formatted: Font: Not Bold
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: (Default) +Body (Calibri), 11 pt, Font color: Text 1, English (United States)
Formatted ...
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)
Formatted: Font: +Body (Calibri)

iteration and 30 independent run

RGA		GSA		PSO		SSA		Proposed Algorithm		Function
Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	
-1.2E+4	-1.2E+4	-2.6E+3	-2.8E+3	-9.8E+3	-9.8E+3	-11760.065	-11175.778	-7.8E+3	-7.8E+3	F8
5.71	5.90	14.42	15.32	56.6	55.1	0	0	0	0	F9
2.16	2.13	6.9E-6	6.9E-6	6.0E-3	9.0E-3	4.44E-16	4.44E-16	4.5E-16	4.5E-16	F10
1.14	1.16	0.04	0.29	0.0081	0.01	0	0	0	0	F11
0.039	0.051	4.2E-13	0.01	0.11	0.29	8.32E-11	4.86E-10	0.00017	0.00022	F12
0.032	0.081	2.3E-32	3.2E-32	2.2E-23	3.1E-18	7.75E-10	8.62E-09	0.00019	0.00205	F13

Table ~~24~~12. Evaluation on multi-modal benchmark functions fixed dimension, 50 search agents, 1000 iteration and 30 independent run

RGA		GSA		PSO		SSA		Proposed Algorithm		Function
Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	
0.998	0.998	2.07	3.70	0.998	0.998	0.99	0.99	0.998	0.998	F14
1.7E-3	4.0E-3	7.4E-4	8E-3	7.1E-4	2.8E-3	0.0003	0.0003	3.7E-4	3.7E-4	F15
-1.0315	-1.0313	-1.0316	-1.0316	-1.0316	-1.0316	-1.031	-1.031	-1.0316	-1.0316	F16
0.3970	0.3996	0.3979	0.3979	0.3979	0.3979	0.397	0.397	0.3978	0.3978	F17
3.0	5.70	3.0	3.0	3.0	3.0	6.599	6.599	2.999	2.999	F18
-3.8628	-3.8627	-3.8628	-3.7357	-3.8628	-3.8628	-3.862	-3.862	-3.8627	-3.8627	F19
-3.3217	-3.3099	-1.9946	-2.0569	-3.2031	-3.2369	-3.32	-3.29	-3.3219	-3.3021	F20
-2.6824	-5.6605	-5.0552	-6.0748	-5.1008	-6.6290	-10.104	-10.104	-5.0427	-6.0489	F21
-10.3932	-7.3421	-10.402	-9.3399	-10.402	-9.1118	-10.17	-10.17	-5.0628	-6.2513	F22
-4.5054	-6.2541	-10.536	-9.4548	-10.536	-9.7634	-10.48	-10.48	-5.1168	-5.50006	F23

Above tables results confirm the high performance of proposed algorithm in high dimension operated better than the others in a substantial number of benchmark functions and have a

relatively good performance in the others. Parameters used in the two tests are provided in A and B appendix.

For more analysis, average data in Table ~~22~~10, ~~23~~11 and Table ~~24~~12 are used to perform non-parametric tests. In this stage, Friedman tests (Friedman, 1937) and (Li, 2008) are performed by platform STAC (Rodríguez-Fdez, 2015) on collected data to form one versus all of the significance level of 0.05 and the their results are exposed in the table ~~25~~ 13 to table ~~27~~15.

Table ~~25~~13. Friedman test result

Result	p-value
H0 Rejected	0.00025

Through rejecting the default hypothesis in table ~~25~~13, Friedman test Result proves that the average results obtained from proposed algorithm are different at least in comparison with the two others. Also ranking obtained from this test (table ~~26~~14) confirm the proposed algorithm as the best algorithm in terms of performance in the estimated optimal solution.

Table ~~26~~14. Ranking result of Friedman test

Ranking	Algorithm
1.82609	Proposed algorithm
2.31349	SSA
2.41304	GSA
2.60870	PSO
3.15217	GA

The results of Lee test shown in Table ~~27~~ 15 confirm Friedman test results by rejecting the default hypothesis in two PSO and GSA and show that the proposed algorithm has different solutions comparing with those two algorithms.

Table ~~27~~15. Li test result

Result	p-value	Comparing with:
H0 Rejected	0.00056	GA
H0 Rejected	0.04342	PSO
H0 Rejected	0.12312	GSA
H0 Rejected	0.11045	SSA

Formatted: Font: 11 pt

Therefore by non-parametric tests it is verified that proposed algorithm has more suitable and better performance comparing with other algorithms in studied cases which is significant in some problems.

5. Discussion

In this article, a meta-heuristic algorithm proposed to solve optimization problems based on swarm intelligence. The proposed algorithm is designed and implemented based on the history of pirates and their life style and their activities over 16 and 17 century its parameters are set intuitively. Search agents which are the pirate ships explore problem space looking for the optimal treasure (point). Pirates tend to reach to a better position by pursuing other ships supposing to improve their position and status in the problem space. Treasure maps are another important and effective factor on the ships movement that point to best ever seen location by the population. Maps that are only the best pirates have access to. The legend operator tries to put together the treasure maps to generate better positions, and the storm operator tries to make a chance of avoiding local optimum by sudden changes in the time. In each round of the algorithm, top ships participating in the battle may obtain a better location in the problem space. Also by giving chance to the looser to re-explore the problem space from a new point. According to transferring information method in the past, the algorithm is designed with a memory include the existing ship status, and each ship is allowed to use only a part of this information i.e. accessing to a parts of memory with respect to its situation.

Finally, the last goal is to reach the optimal position by following the best pirate, Captain Every, by ships who know his location. Due to their status, ships set their own sail to achieve the right speed for exploration or exploitation in the problem space. Besides the sail, ship fitness representing treasure of that ship, has a direct impact on the ship's weight and leads the higher fitness ship to have more treasure and consequently more weight. Therefore it has less speed than other ships to be able to properly exploit nearby the estimated optimal position. Wind, as a form of adjusted parameter, is reduced during the implementation of the algorithm in order to search more accurately with smaller steps at final stages.

By solving a sample problem, status of the ship's movement are observed in different stages. These ships converged at a point where imagined as the global optimal. Finally it is observed that the algorithm has the ability to explore and exploit by testing the performance of the algorithm on 23 high usage benchmark functions so it is able to improve the time of the best solution, and enhance the status of population fitness gradually. In continue, it is found that the algorithm is relatively good at finding the optimal solution in most of studied functions and revealed a better performance comparing with other algorithms in a significant number of these benchmark functions and is able to solve a wide range of optimization problem by adapting itself. Then, by examining the performance status of the algorithm in higher dimensions, we found that the algorithm has an ability to generate a suitable solution in the problems with a significant number of inputs.

6. Conclusion

In the proposed algorithm, search agents are considered pirate ships looking for best fitness positions by moving in problem space as parts of the sea. To gain more wealth, they tend to identify and move towards greater wealthy ships. The amount of ship wealth is determined by the location

of that ship. This amount of wealth and fitness has a direct impact on making decisions for speed control and ship movement in the next step. Wind interference and some creative changes make search agents as natural as possible for simulating ships movement. Standard evaluation functions such as Rastrigin, Rosenbrock, Griewank and Zakharov are used to evaluate the algorithm. Algorithms such as GA, PSO, and GSA are used to evaluate and compare with proposed algorithm. Parameters of these algorithms are chosen in the best possible ever state by try and error in order to catch their best performance to solve the evaluation functions. Each algorithm run 25 times to increase the ability of the results confidence and then the average results use to measure their efficiency. Obtained results shows that the algorithm is able to explore and exploit in search space as expected and also has the ability of improving the average fitness of population and improving the best so far solution.

~~**Future work**~~

Declarations

Funding: There is no funding for this paper.

Conflict of interests: There is no conflict of interests between authors of this article.

Availability of data and material: There is no data for this paper.

Availability of code: Code is available by request.

References

Atashpaz-Gargari, E. & Lucas, C., 2007. *Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition.* s.l., s.n.

Baugean, J., 1826. *Xebec Spanish.* [Online] Available at: https://commons.wikimedia.org/wiki/File:Chebec_espagnol_en_1826.jpg

Boddy-Evans, A., 2018. *The Trans-Atlantic Slave Trade.* [Online] Available at: https://www.thoughtco.com/the-trans-atlantic-slave-trade-44544

Chou, Y.-H., Kuo, S.-Y., Yang, L.-S. & Yang, C.-Y., 2018. *Next generation meta-heuristic: Jaguar algorithm.* s.l.:IEEE Access.

Eberhart, R. C. & Kennedy, J., 1995. *Particle swarm optimization.* s.l., Proceedings of IEEE International Conference on Neural Networks 4.

Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *American Statistical Association,* Issue 32 , p. 674–701.

Geem, Z. W. & Kim, J. H., 2001. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation.*

Goldberg, D. & Holland, J., 1988. *Genetic algorithms and machine learning.* s.l.:Machine Learning.

Heller-Roazen, D., 2009. *The Enemy of All: Piracy and the Law of Nations.* s.l.:s.n.

Koziel, S. & Yang, X.-S., 2011. *Computational Optimization, Methods and Algorithms, Studies in Computational Intelligence.* s.l.:Springer.

Li, J., 2008. A two-step rejection procedure for testing multiple hypotheses. *Statistical Planning and Inference,* Issue 138 , p. 1521–1527.

Mavrovouniotis, M., Li, C. & Yang, S., 2016. A survey of swarm intelligence for dynamic

optimization: algorithms andapplications. *Swarm and Evolutionary Computation.*

Mirjalili, S., 2014. Grey Wolf Optimizer. *Advances in Engineering Software,* Volume 69, p. 46–61.

Mirjalili, S., 2015. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput & Applic.*

Mirjalili, S. & et al, 2017. Salp Swarm Algorithm: A bio-inspired optimizer for engineering. *Advances in Engineering Software,* pp. 0-29.

Mishra, A., Das, M. N. & Panda, T. C., 2013. Swarm Intelligence Optimization: Editorial Survey. *International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 1.*

Rashedi, E. & Nezamabadi pour, H., 2009. GSA: A Gravitational Search Algorithm. *Information Sciences,* 179(13), pp. 2232-2248.

Reid, S., 2015. *Fitness Landscape Analysis for Computational Finance.* [Online] Available at: http://www.turingfinance.com/fitness-landscape-analysis-for-computational-finance/

Rodríguez-Fdez, A. C. M. M. A. B., 2015. STAC: a web platform for the comparison of algorithms using statistical tests. *Proceedings of the 2015 IEEE International Conference on Fuzzy Systems.*

Sid meier's pirates. 2004. [Film] Directed by Sidney K. Meier. s.l.: 2K Games, Atari.

Sörensen, K., 2015. *Metaheuristics the metaphor exposed.* s.l.:International Transactions in Operational Research 22.

Xue, J.K. & Shen, B. 2020. A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm. *Systems Science & Control Engineering*, 8, pp 22–34.

Xin-She Y., & Xingshi H. 2009. Firefly Algorithms for Multimodal Optimization Proceedings of the 5th international conference on Stochastic algorithms: foundations and applications, pp 169–178

Formatted: Font: (Default) Times New Roman, 12 pt, Not Italic

Formatted: Font: Times New Roman, 12 pt

Formatted: Normal, Indent: First line: 0.5 cm

Formatted: Font: (Default) Times New Roman, 12 pt, Font color: Auto, Do not check spelling or grammar, Pattern: Clear

Formatted: Font: (Default) Times New Roman, 12 pt, Font color: Auto, Do not check spelling or grammar, Pattern: Clear

Formatted: Check spelling and grammar

Formatted: Font: (Default) Times New Roman, 12 pt, Do not check spelling or grammar

Formatted: Font: (Default) Times New Roman, 12 pt, Do not check spelling or grammar

Formatted: Font: (Default) Times New Roman, 12 pt, Do not check spelling or grammar

Formatted: Font: (Default) Times New Roman, 12 pt, Do not check spelling or grammar

Formatted: Font: (Default) Times New Roman, 12 pt, Not Bold

Formatted: Font: (Default) Times New Roman, 12 pt, Font color: Auto, Do not check spelling or grammar, Pattern: Clear