Sea Lion Optimization Algorithm for Solving the Maximum Flow Problem

Dr. Nidhal Kamel Taha El-Omari

Department of Software Engineering, Faculty of Information Technology,
The World Islamic Sciences and Education (WISE) University, Amman – Jordan

Summary
There is a shred of ample evidence that optimization is an enormous field that pervades essentially every aspect of our day-to-day life ranging from academic and engineering fields, going to industrial and agricultural segments, passing through social domains, and ending with commercial and business sectors. Evidently, the philosophy of optimization has emerged out of the utmost need for finding the best available solution among a set of candidate ones, without which our life will lose its vitality.

Over the last few decades, a worthy amount of interest has been focused on finding solutions for a wide range of intractable optimization problems by scientists and researchers from diversified domains not only for academic and research objectives but also due to the existence of a wide variety of real-life applications. They indeed see the remarkable resemblance between the swarms, for instance, and the behavior of a human in solving problems and trying to come up with new goal-oriented operating methods to tackle many important real-world problems. Nature Inspired Computing (NIC), as its name implies, is the fusion of nature, by itself, and Artificial Intelligence (AI) to solve various global optimization problems. Furthermore, swarm optimization is considered as the most representative of these nature-inspired algorithms. Motivated by applying natural phenomena to metaheuristics and trying to simulate the harmonious behaviors of creatures in solving problems particularly the joint hunting behavior of the sea lions, the aim of the research work reported in this paper is twofold. On the one hand, many theoretical and practical aspects of heuristic and metaheuristic approaches, from classical to novel approaches, are discussed and covered. On the other hand, this nature-inspired paper addresses a pioneer metaheuristic optimization algorithm in the context of finding the optimal solution for the Maximum Flow Problem (MFP). To be more precise, this paper elaborates on using the Sea Lion Optimization (SLnO) Algorithm for solving the Maximum Flow Problem (MFP), hence the name "SLnO-MFP".

After the proposed solution SLnO-MFP algorithm is analyzed and the experimental tests are conducted on various real-case datasets, the reported practical results are represented, discussed, and compared using the same datasets with other algorithms, including the Whale Optimization Algorithm (WOA) and Ford-Fulkerson (FF) algorithm, which have been used to solve the same problem of interest. As the accomplishment achieved in this valuable research is efficient and robust, the proposed algorithm is proved to be a senior-level alternative to the optimization problem and, in turn, can be efficiently used to solve various optimization problems having a fairly large-scale data such as the underlying problem (i.e. MFP).

Keywords:
Artificial Intelligence (AI), Artificial Neural Network (ANN), Global optimization, Maximum Flow Problem (MFP), Metaheuristic Algorithms, Optimization, Sea Lion Optimization (SLnO) Algorithm, Swarm Intelligence.

1. Introduction

Every aspect, visible or invisible, of our life, encompasses inside its folds a wide range of optimization problems. Not just that, every real-world system, or even a portion of a system, can be abstracted as an optimization system and encapsulates internally one or more optimization problems. In the most basic sense, the primary interests of optimization algorithms (OAs) are pivoting around making something more and more effective to the greatest possible extent by recursively searching to provide more refined and scalable solutions for the problem of interest [1][2][3]. This implies that there is an ultimate need for reformulating the considered problems in terms of optimization which, in turn, has two faces of the same coin: first, generating a set of candidate solutions for the desired problem which is referred to as the "search space", second and more importantly, assessing the achieved performance of these available solutions based on some quality measures which should be previously defined [4][3][5]. These quality-measures or as so-called fitness functions, objective functions, or goodness levels are quantifiable and revolve around maximizing some desired features and/or minimizing the undesirable ones until a predefined optimization goal is achieved [6]. Generally speaking, neither generating these solutions from the search pool nor devising their objective functions is a simple task to do; hence, most real-life optimization problems are too challenging to solve [4]. Inevitably, there is a pressing necessity for a search methodology that is used to get in-depth information that originally exists in one way or another within the promising search space of the problem domain. In terms of this, there are broad ranges of searching algorithms and a number of features for which to categorize them. Some are classified according to the searching strategies, some are classified by the searching scope and area, some are classified upon the optimality of their output, some are classified by their ways in generating

Manuscript received January 3, 2020
Manuscript revised August 25, 2020

DOI: 10.22937/IJCSNS.2020.20.08.5

solutions, and so forth. Each of them is inspired, implicitly or explicitly, by an existing natural world phenomenon or a certain sort of metaphor [6]. Nonetheless, all of them seek around the same goal of improving effectiveness.

Broadly speaking, there are actually many types of networks that are routinely faced through daily life-cycle. These various types, which have enormous practical importance in our life, include the following real-life examples: Internet, telephone, cell, highways, rail, water, sewer, electrical power, oil, and gas, to name just a few. Depending on each one, every network has a material flowing from point to point [7][8]. While each point is referred to as a node, each connected path between any two nodes is called a route or an arc [7][8]. In analogous to its type, each material has a corresponding unit to measure the flowing capacity such as time, price, distance, quantity, and other units [7][9]. Based on this, optimization is a solution procedure in which one aims to systematically enhance the material flowing through a given network [4]. From a different perspective, this enhancement can be either maximize the goodness or minimize the badness of a stated solution [4]. Without loss of generality, the maximization problems are considered instead of the minimization throughout this paper. In case of the need for considering minimization problems, the same methodology is simply used after reversing the sign of calculation. To this objective, this paper is trying to reach the maximum flow capacity of the network at which the flow can be transmitted more reliably from the starting source node "***s***" to the target node "***t***". This rate is greatly related to the same network "***G***", which, undoubtedly, depends heavily on both the number of nodes "***n***" and the number of edges "***m***" [7][8]. Going forward, this problem itself is known as the Maximum Flow Problem (MFP) where the search space is actually represented by a graph, an example of which is shown in Fig. 1 where the first value that is associated with every edge represents the actual flow value and the second value represents the maximum capacity value; this will be explained later. The comparison of these networks are often defined by the amount of flow, the number of edges and vertices they have, as well as whether their flows are bidirectional or not. The values of "***n***" and "***m***" of this example are equal to seven and ten, respectively. It is worth remarking that neither side of "***m***" and "***n***" is dominant over the other, both are important in driving the size of the problem. In this regard, both the number of nodes "***n***" and the number of edges "***m***" with their capacities determine the complexity of the network "***G***" and, for that the runtime complexity associated with any instance generated from "***G***" increases rapidly with the dimension of the network graph, i.e. the numbers of vertices and edges [10][7].

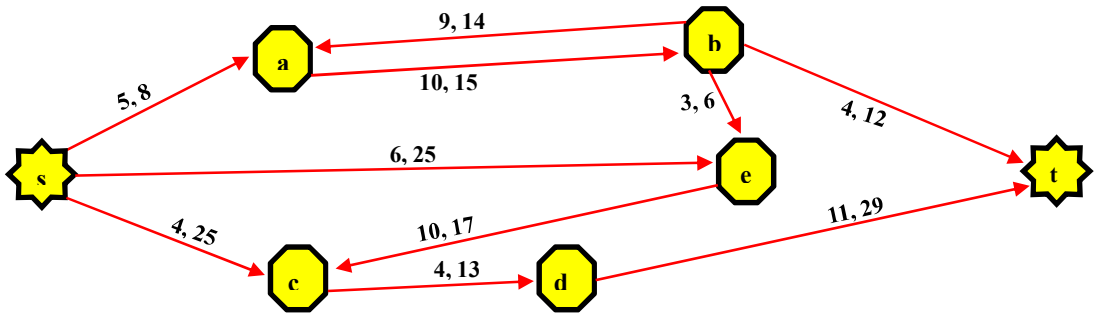

Fig. 1. An example of MFP, having seven nodes and ten edges

As shown in Fig. 1, MFP is much related to the essence of the objects' movement through the desired network where four milestones are there: source place called a source or a start node "***s***", a destination node called "***t***", one or more connected routes between the source and the target, and an associated positive number to represent the flow between every directed route that connects any two nodes [11][8]. In order to simplify the problem of interest, one can think of this problem as using pipes of different sizes for carrying liquid from a source node to a destination one. Or, one can think of this problem as using conduits to link between the start and the destination nodes. Anyway, the important thing is the availability of several intermediate connecting paths, called by routes or tracks, which can be followed in carrying the flow between "***s***" and "***t***". It is vitally important to mention that MFPs are part of the graph paradigm that doesn't require capturing every low detail of the problem under discussion. The rationale behind this relates to the fact that they are mainly based on abstract concepts where deep knowledge of the problem at hand is no longer required. And so, the ordinary user can use the metaheuristic optimization algorithm in solving many optimization problems without having to have an in-depth exhaustive understanding of the same algorithms.

The critical thing to keep an eye on is that nodes can't transfer any matter more than their previously defined capacity value [11][7]. This term might be generalized in the long-run as that any network can't transfer more than its computed capacity value. Whenever there is more than one feasible track to be chosen between the source and the destination nodes, the MFP is regarded as a player with a considerable role in enabling a greater movement of these objects or matters. And so, the intended objective is to choose the most optimal route between the source "***s***" and the destination "***t***" that has the maximum throughput, i.e. flow [7]. Even though the focus view varied depending on each network, one fundamental three-sided cornerstone remained fairly viewed in a broad sense: speed up the flow to the looked-for value and reduce the time and cost. Simply, this objective can be redefined as the more flow the network has, the more efficient and effective it will be.

To this aim, the core motivation behind this research article is to get the maximum flow capacity at which this flow might be transferred reliably between the two special extremes: the source node "***s***" and the destination node "***t***". However, this amount is closely associated to the same network "***G***", which, in turn, rely upon the various intermediate paths between these two extremes and on both the number of nodes "***n***" and the number of edges "***m***" by which the size of the network is defined accordingly.

This research is concerned with solving optimization problems and describes a new metaheuristic optimization algorithm (namely, Sea Lion Optimization Algorithm for Solving the Maximum Flow Problem, SLnO-MFP) which, as a member of the swarm-intelligence family, mimicking the hunting behavior of the sea lions. According to the acquaintance of the author of this research, there is no previous literature on the usage of SLnO for solving the MFP and this finding has not been reported yet in the optimization literature.

In line with the said objectives and in order to lay the foundation of this paper, an outline of this paper is structured as follows. After this section justifies the importance of this research and provides background knowledge on the maximum flow problem for the novice readers, Section 2 explores the depth of literature to introduce both the heuristic and the metaheuristic optimization paradigm. Section 3 surveys the literature within the research area to gain a concise overview of the other related work and, moreover, it ferrets out the objectives of the underlying problem and discusses some of the different algorithms which are proposed to solve this problem intelligently. The model formulation and the different assumptions related to the problem of interest are provided and discussed in Section 4. In order to build an authentic depiction of the considered problem, the formulation of the theoretical and mathematical foundation of this proposed solution is introduced in Section 5. Section 6 is where the real work begins; it takes a closer look at the current algorithm developed in this contribution and then walks through all the various stages which would be required to implement. While the conducted experiments and their detailed intensive analysis are discussed in Section 7, Section 8 concludes the project work of this research. Lastly, to close the discussion of this research article, Section 9 offers some ample research scopes and introduces a fairly wide range of promising research opportunities in furthering the aims and objectives of the research.

2. Background of Metaheuristic Optimization Algorithms

With the purpose of providing a self-explanatory paper, this section establishes preliminary knowledge of the background pertaining to the concepts of the optimization paradigm and draws an inclusive image for both the current and future status of metaheuristic research. Therefore, the following subsections discuss the different categories of optimization algorithms.

2.1 Types of Searching Algorithms

As such, there is definitely a broad range of techniques proposed in today's progressive and growing arena of optimization; each of which has its own capability, strength, weakness, objective space, detailed specifications, constraints, requirements, and other fundamental relevant features of searching. Since most of them claim a progressive style to implement, the right decision for the most reliable algorithm to encompass a given problem selection is no longer a simple mission to be carried out. Bearing in mind that most of the complicated hard problems can be framed within the optimization borders in which one strives harder in an effort to either minimize or maximize the achieved results within limited resources [12][13], Fig. 2 is directly interrelated to the question of which algorithm to choose and which notions to implement for a given problem of interest. In terms of reaching the most right one of the optimization algorithms, this figure deliberate and then formulates the different critical factors which influence people's decisions in picking up one of them. However, there are various distinct elements between them which overlap with each other. Viewed in a broad sense, the classification of these algorithms encloses, but is not limited to, the following overlapped categories:

- **Problem Functionality:** In terms of functionality, algorithms can be classified into two major types: problem-specific and problem-independent. While the former provides that the algorithm, as well as the code, is just tailor-made for a specific type of problems, the latter assures that the same code is used for various varieties of problems, namely it is a generic-implementation code. While the focus of the latter search strategies is mainly concentrated on using the algorithms without any problem-dependent knowledge, the former look as if they were tailor-made for a narrow range of problems. [14]
- **Searching Scope:** From a classification point of view, the searching scope is used as well. Different than the global ones that are commonly used in finding the global optimum quality solutions, local search techniques might become stuck in the so-called local-optimum values of the solution space when no better solution is observed in the present existing neighborhoods. Even though both techniques are striving to find a solution that more optimizes the cost criterion among a set of candidate ones, the distinction between them is that the global search looks at the entire problem space as a single entity when trying to find the best possible candidate solution [15]. Despite that all of the global optimum solutions are never guaranteed concerning solution level of quality, finding global optimums in global search ones could be more counted on.

  In the pursuit of being on the best of what it has already been achieved so far, most current-state-of-the-art optimization algorithms fall within the realm of iteratively improving the outcomes in the course of the search process. Thus, by considering their ways of generating solutions, further additional taxonomy can be possible for

the exploration of the solution space effectively. Fig. 2 exhibits that the searching process falls into quadruple pivot points: stochastic, exploration versus exploitation, iterated (i.e. iterative), and guided:

- The searching has a stochastic nature when the set of random variables is employed in extracting a new generation, called a potential solution [9][16]. Each generation's new values of these variables are chosen stochastically in harmony with the general paradigm. On the other side, some problems are evolving stochastically at different points in time which makes the optimization hard to grasp and solve. For instance, the passengers' numbers of airlines occur stochastically which calls the airlines for implementing the statistical theories, including stochastic analysis and probability distributions, in forecasting the numbers of passengers.
- Exploration and exploitation are two supplementary activities used to explore the search space. The first one is the activity through which the search algorithm tries to explore as much broad search space as possible to evade falling in the local-optima traps. Whereas the second one is representing the activity in which the searching algorithm tries to develop the finest discovered solutions through some targeted approaches. While exploitation is the optimized outcome derived out of exploration, the progress of the search for more solutions continues in both cases to find more-optimal ones. [10][17][18]
- For highly efficient exploiting the former iteration (also called trial or time-step) to the greatest possible extent, the outcomes in the guided search are frequently improved over the course of the iteration steps where the newly generated trials are influenced by the older ones. The basic idea of that is wrapping around deliberating knowledge and extracting patterns from the former good-quality searching iterations in an effort to harmony guide the searching process in the subsequent iteration steps hoping to be in the optimal solution direction, hence the notion of “guided” is used to continuously approximate the goal based on replacing the old solutions with the new successful ones. From a more general angle, the useful information related to the optimas of the preceding iteration steps is stored on to get benefit from them in the succeeding ones. That is, the key philosophy of deriving more successful trials is coupled with building up a well-stocked knowledge store containing the past trials. [10][17][16]

  Unlike the conventional local search that stops when it gets stuck by any local optima, the guided local search makes the best use of the available features of the current optima to escape away and then form another more optimum feasible solution. In the guided search absence, an optimization algorithm (OA) inevitably takes on a longer exploratory time range that is mainly driven by the trial-and-error aspects. [10][17][18]
- The progress of the searching process will be continuing iteratively with the same searching procedure until one or more of the predefined evaluation functions, called termination conditions or stopping criteria, imposed by the user are reached, and accordingly, the best possible candidate solution is produced. Without that, the progress of the searching mechanism will obviously be in an infinite loop [10][17]. The followings are some of the possible termination conditions that may be imposed by the user to terminate the series of iterations: the maximum number of iterations steps previously defined (i.e. no. of runs) is reached, the maximum allowable CPU computational run-time (i.e. max-CPU-time), coming across some significant evidence that an optimal solution has been achieved, the maximum number of iteration attempts that comes amid two successive developments is reached, the maximum number of iteration attempts has reached without noticing any difference or making any forward positive progress for the problem of interest, or there is no way to get more developments for the problem of interest (i.e. there is no progress) [10][17]. Related to the first termination condition, the applications of the Artificial Neural Networks (ANNs) use the name “epoch” to represent an iteration step (i.e. time-step) [10][18]. In-line with the second termination condition, it's very crucial to balance between the quality of given feasible solutions and the overall computation time frame that is utilized in generating these solutions and decide accordingly how to set up the timeout bound [10][18]. On the other hand, the last termination condition represents the case that after many generations, the solutions start approaching each other in the hope that this approaching is a good indicator that the final achieved solution is closer to the ideal solution of the problem under research [10][18]. In the report of this, there is an essential need for suitable criteria to define the quality of the acceptable solution and to decide according to that whether to stop the searching activity or not [10][17][18]. If the procedure fails to reach a visible solution or a practical compromise within the timeout bound, it is inevitably stopped [10][18].

- **Output Optimality:** The problems are basically categorized into two different types of models: Deterministic (i.e. exact) and Stochastic [9]. While the first one is directly associated with the problems in which the different-used variables are known in advance with certainty and before solving them, the second model is indicating to the cases where the associated variables involved a degree of uncertainty [9][18]. Upon the optimality of their output through the different runs, algorithms are generally categorized into two fields: deterministic and nondeterministic. Within this context, the

"Deterministic Algorithm", also known as exact algorithms, describes the cases where the same algorithm at all times of the repeated runs will definitely produce one and only one same output, called solution, for the given particular input data. As this grip on the reality of a single certain outcome, it is obvious that their exclusive orphan solution is the provably optimal one, and nothing else. Furthermore, since such solutions can be extracted within a fair rational time, they are used only for problems of small-scale instances as they defined under the term "Deterministic". Conversely, complex large-scale instances can't be generally resolved within a rational time by using these classical exact approaches. This is caused by the fact that proceeding ahead in the real-world with all possible solutions, as is the case of exact or precise approaches, is sometimes time-consuming and hard to sustain in terms of resource availability and utilization. From a computational aspect, enumerating and checking all potential candidate solutions for optimality satisfaction is systematically impossible for the majority of large-scale optimization problems especially those involve hundreds and even thousands of variables. [19][20]

Going forward, the "non-deterministic algorithm", by contrast, means that the same algorithm may show some alternative behaviors from run to run even though the input data are the same. Form another perspective, this discrepancy in the behavior is attributed to the fact that the calculation is subject to some norm of randomization and, for that, the outputs have a fluctuating-stochastically behavior and may vary from one run to another. With regard to this norm of uncertainly, all the generated behaviors are considered as valid outcomes and every one of them may be the optimal solution or close to the sole optimal one. And so, every execution for any algorithm belonged to this type hides a degree of "uncertainty" or "randomness" behind its output. Definitely, this extent of "uncertainty" is only limited to agreed sets of rules that should be defined before. [9][17][21][5]

Beyond that, this category is commonly used when the tackled problem tolerates multiple possible outcomes where all of them are considered as valid ones through the solution space without providing proof of optimality. Within the fact that they may perform differently within various routes, the non-deterministic algorithms are extensively used in finding estimated solutions; this is specifically true when we coming across credible evidence that revealing the optimal solution among a set of possible candidate ones is beyond the capability of exact algorithms and, perhaps more importantly, an optimal solution is too costly to be attained especially in terms of time function. [14][5]

- **Time complexity:** By considering their time-growth complexity, a further taxonomy can be possible for these algorithms; Fig. 2 illustrates that algorithms can be forked into two subfields: polynomial and non-polynomial. Polynomial, as the name implies, means that the tackled problem can be initially solved from scratch in polynomial run-time by not less than one algorithm [10]. However, in the other case, the problem at hand needs non-polynomial time to be solved which often means too long computational time to be tolerated [10]. If the processing time is narrow and usually it is that, there is a sorely need to sacrifice the seeking of the solution optimality at the expense of near-optimal solutions [20].

- **Problem Hardness:** When the last two categorizations are integrated together, another categorization is emerged out of them. In view of this, the problems themselves can be categorized according to their complexity: Polynomial problems and NP-hard problems. Even though the former one is directly related to the problems whose computational time is growing polynomially with the problem size, the latter is relevant to the problems whose time-growth rates are often growing exponentially with the size of the problem. Notwithstanding that the computational time of the NP-hard problems might not strictly with exponential increases in all cases, but they are definitely not polynomially. As opposite to NP-hard problems, the time of the former category is firmly constrained by a polynomial function based mainly on the problem size. For instance, suppose that "**q**" is the problem size, then all the followings are polynomial functions "$\mathbf{q^2}$", "$\mathbf{q^3}$", "$\mathbf{q^4}$", "$\mathbf{q^5}$", etc. Quite the opposite, there isn't any known polynomial algorithm that is capable to solve the problems that lie under the latter category. As a matter of fact, most optimization problems are classified under the second category and they are describable as non-deterministic polynomial-time hardness (NP-hard) problems which address the case that a solution for the problem under consideration can be achieved within a polynomial time by using a nondeterministic computer without providing proof of optimality. [22][23][13][10]

  The followings are some of the key factors that are related to NP-hard problems [22][10]:

  - It is often the case that most of the NP-Hard problems are very easy to define and described but so hard to be framed or/and solved as optimization problems.
  - Searching space: These problems are usually of huge dimensions, namely, they involve a fairly wide range of possible solutions so that they are usually very hard to be tackled.
  - Solution's quality level: The good-quality assurance or excellence of the calculated results is not guaranteed. But, if the optimal solution has not been reached, it doesn't mean that a "good" one isn't achieved. All in all, this is much related to the nature of the underlying problem. In terms of time-growth complexity, the perceived relationship between the optimal solution (i.e. the best candidate solution) and the size of the problem under consideration is exponential. As soon as the problem size begins to mount, the computational time needed for further refining the candidate solutions

is growing at an exponential pace. In such scenarios, these algorithms need a relatively long processing time for driving the optimal solution and, as a result, these problems are generally not resolvable within a rational amount of computational time. In a variety of cases, extracting an approximation to the optimal solution is also hard to be achieved within a rational period.

- Exhaustive search: In practice, brute-force examining of all candidate solutions may be placed into the realm of the sheer impossibility.
- Time-growth complexity: Since this norm of problems is ordinarily large-scale and, as a result, demanding some “expensive” time computations, they are often difficult to solve. To this end, these problems are generally calling for exponential resources to reach the optimum quality solution or the near-optimum one.
- Despite the fact that some of these problems are said to as having a polynomial-time algorithm to solve but as a matter of fact, no anyone at all sets apart what the algorithm is!

In all these contexts, these problems are also referred to as long-term problems. To state a truth, the largest fraction of real-world optimization problems falls into the NP-hard class or at least in the sense of NP-hard. [22][10]

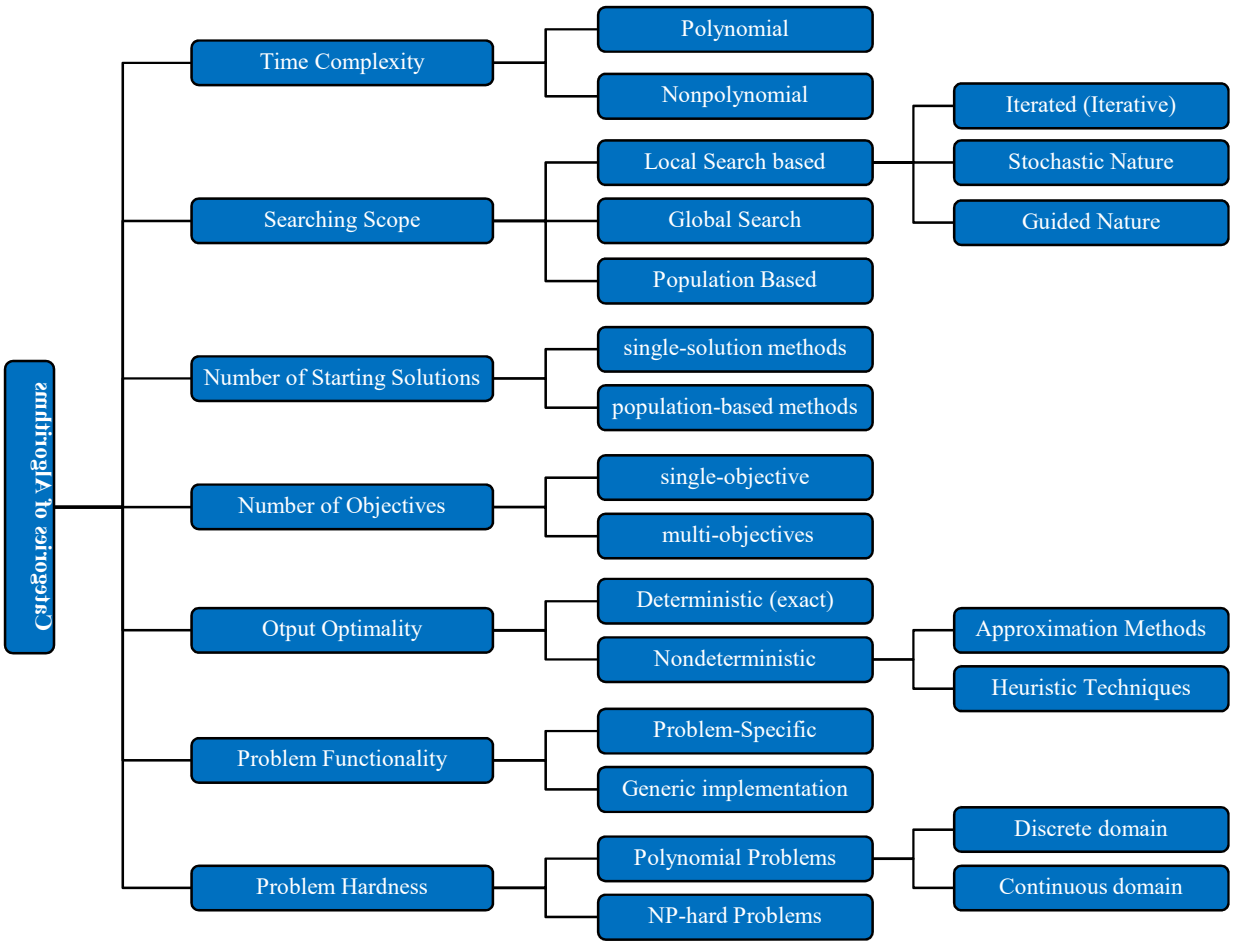

Fig. 2. Types of Searching Algorithms

- **The Number of Objectives:** Depending on the nature of the problem, some problems have a single-objective function while there are many others that have goals with multi-objective functions, referred to as multi-objective optimization. In reality, the latter case requires to be incorporated with a weighted average to reflect the nature of the several objectives' existence. So, a multi-parameter vector is used for their fitness functions. [6]
- **The Number of Starting Solutions:** On conformity with the problem domain and in order to come up with these classifications, single-solution (also referred to as trajectory) versus population-based searches may be considered as an additional alternative classification element. The following core points are listed here to compare and contrast the two searching strategies:

- The single-solution category contains methods that start by choosing one solution randomly and then enhanced it in the course of the search process. Since these methods contain only one solution in every one of the iterations, they are also called single-point or trajectory methods. Simulated Annealing (SA) and Tabu Search (TS) methods are the most leading examples of this category On the contrary of starting with a single nominal solution, the population-based category starts initially by generating a set of multiple random solutions, and then these solutions are enhanced extensively towards more superior search areas throughout the series of iteration steps. The enhancement of the population-based strategy is emanated either by recombination of more than one solution into a single one or reforming each solution by the use of a given strategy adopted especially to impose exploration and exploitation of the search space. [10] [20][24][16][18]
- A higher exploration power is attained in the population-based towards finding out the overall global solution rather than staying on local ones. On the other hand, the nature of the single-solution category is considered as more exploitation oriented. [20][24][16]
- Since the abstracted knowledge about the search space in the population-based approaches is shared between many possible solutions, there may a sudden and widespread shift in the direction of the optimal solution [16].
- The recombined solutions of the population-based methods are normally based on big-guided steps while these steps in the single-solution methods are commonly smaller-guided and, of course, the movement of each of the two alternatives for more productive solutions and bettered outcomes is only within its corresponding own search space. Despite these solutions' improvements, these big and small guided steps are at the expense of the danger of being close to or missing good solutions where this danger is higher in the population-based approaches of that of the single-solution approaches.[10][16]
- By the population-based methods, multiple possible solutions collaborate with one another to go beyond local-optima traps [16]. Namely, all new solutions are built on previous ones and provide inspiration for future ones.

With respect to the fact that there are no clear-cut boundaries that are determining where the above-named categories start and stop, there might be many varieties of hybridized categories. For instance, a further forked group may compose of some approaches which deal directly or implicitly with the graphs. From a different point of view, the positive capabilities of two or more approaches may be fused together to form a new hybrid technique that can be used to solve problems from the same domains or the others.

2.2 Approximation and heuristic approaches

Due to the stated limitations of the classical exact approaches in supporting most complex optimization problems, scientists and experts from both research and industrial communities think intensely for finding possible alternative approaches that are developed to support and capture efficiently the solution of the optimization problems within a fully acceptable time border even if there aren't some high levels of certainty. To this end, approximation and heuristic approaches are eventually evolved for finding the optimum or at least close-to-optimum solutions regardless that these approaches have no assurance for the computational time or the accuracy of the in reaching the optimal solution. Ground truth, the quality level of the approximated methods is ordinarily under the terms of predefined boundaries that are not far off the exact solutions. In contrast to this, the quality level of the heuristics methods is not guaranteed in exploring the global optimum solutions or to be within these predefined boundaries; however, the exact results might be caught in some exceptional situations. Unlike the approximate ones, heuristic approaches may have included some chancy errors that are incapable of being anticipated. [25] [1][14][10][20]

To further control optimization problems, there is an utmost need for a higher-level of heuristic especially when one be faced with some extensive searches that have one or more of the following feasible constraints and obstacles: [14][26][13] [9][27]

- **Information constraint:** Incomplete, limited, imperfect, or conflicted pieces of information upcoming from different causes and sources.
- **Resources constraints:** Restricted by limited computation capacity or with resource availability and utilization.
- **Time constraint:** Guaranteeing the computational time to be within the stipulated time is an ever-growing concern for the decision-makers and all the stakeholders in both the industry and academia communities.
- **Problem difficulty:** The tackling problem is, to some extent, a difficult optimization one that is comparatively hard to solve.
- **Quality constraints:** In some occasional cases, the search process may be caught by some local-optima traps without having the ability to bypass them. However, it is an important issue to look beyond these local optimas in the hoping of finding the global optima.
- **Knowledge constraint:** A shortage of sufficient knowledge to design the equivalent well-organized solving methods.

Crucially, all of the above-stated critical issues are worthy enough to address the necessity for "high-level heuristics" especially for the cases where capturing every low-level detail of the considered problem is hard to attain in a reasonable amount of time. Due to these challenges and with the advancement in alternative modeling, scientists over the past few years are constantly trying their best to come up with new goal-oriented operating methods to solve these important real-world issues. They find their enlightenment and guidance by abstracting the structure and function of nature's laws, by itself, and the so remarkable behaviors of the different creatures in solving problems; hence, "metaheuristic" algorithms arose into the vision among which nature-inspired algorithms are actually the largest fraction of them. The next subsection introduces the basic concepts of metaheuristic algorithms. [10][2][28][16]

2.3 Metaheuristic

Fig. 3 illustrates the most ten leading areas that are imitated by most metaheuristic algorithms. It is obvious from this figure that insects are the most popular imitated area among these areas where (23%) of the total publishing metaheuristic literature are concentrated on mimicking the living ways and the survival systems of insects. The next area has (17%) which is inspired by the natural evolution of Darwin's theory of evolution and survival (i.e. survival-of-the fittest). Then, the next one is with animals (whales, wolves, fish, cats, monkeys, bats, and many others) which has (16%), and so on up to the percentage (4%). To state a relevant truth, the social behavior of bees followed by ants are the most top favorite insects that are foremost imitated and reported while searching the related metaheuristic literature. [20][27][29]

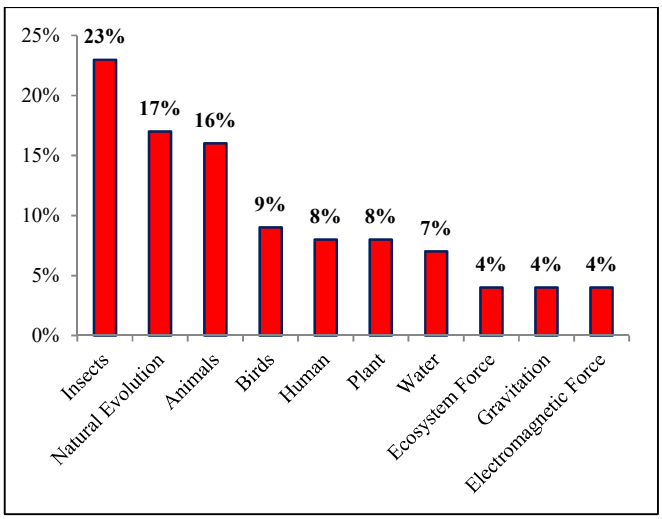

Fig. 3. The top ten leading metaheuristic areas

On the other hand, the drawing of Fig. 4 states that (93%) of the available reported metaheuristics are distributed among six disciplines where more than half of them are classified as nature-inspired optimization algorithms; they are also termed as bio-inspired or bio-based metaheuristic [20][27][29].

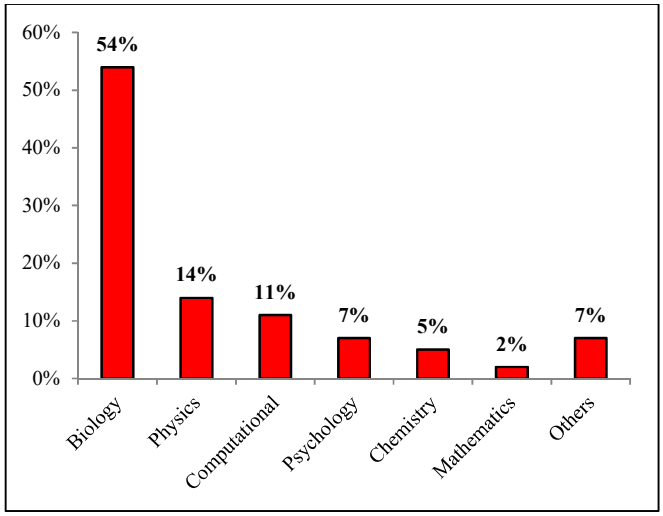

Fig. 4. The top six leading metaheuristics disciplines

In addition to that these nature-inspired optimization algorithms are relatively easier to implement as compared to the conventional optimization techniques used earlier, they can be adopted and implemented in widely varied fields of problems covering multidisciplinary fields and objectives. Above and beyond that these optimization algorithms have more abstract concepts and relying on the usage of simple concepts of the higher-level strategies (hence the term "meta"), they are heuristic, stochastic in their nature, and, perhaps more importantly, they are categorized under the iterative optimization techniques. Besides that they encompass highly-scalable intelligent methodologies and problem-independent algorithmic frameworks, they normally revolve around adding flexibility to the ways of utilizing control parameters that can be customized and tuned to well suit the nature of the problem under consideration. It is worthwhile considering that these techniques can eventually be implemented so that the complex working details are simply abstracted away from the end-users [21]. This high reliability and simplicity that metaheuristics offer are the principle behind their broad diffusion and finding them in numerous successful applications. [10][28][16]

Even though the global optimality of the final metaheuristic solutions among the multiple possible alternatives is not guaranteed or proven to be optimal, these techniques may be at least worthy enough to be trusted in extracting the approximated solutions within reasonable computational time. In its absence, many problems that may be solved with metaheuristics will be inevitably unsolvable. This is especially true for the hard problems in which their exact solutions are too hard to be achieved within rational computation time. As a matter of fact, the price to be paid for the time complexity or as so-named scalability improvement is mainly at the expense of approximation of the optimal matching and, therefore, a fair balancing as empirical as possible between time and quality is definitely a determining factor and a radical issue. [10][20] [28]

Table I highlight the abovementioned notions related to metaheuristic optimization techniques. As they are coined to utilize the power of nature, the source of inspiration for every metaheuristic algorithm has an attractive story behind. Motivated by this and as shown in Fig. 5, they can be categorized upon their common features into at least nine basic categories: [14][15][20][27][2][28][30][17][31][26]

- **Evolution-inspired algorithms:** These algorithms attempt to imitate the rules and laws of the natural evolution of the biological world. Regardless of their nature, these evolutionary-based optimization algorithms are regarded as generic population-based metaheuristic algorithms. The search process of this norm of algorithms has two focal stages; exploration and exploitation. The exploration phase precedes the exploitation phase which can be regarded as the process of exploring in detail the search space. At the exploration stage, the progress of the search process is launched with a randomly generated population which is then evolved over a number of subsequent generations. The most applicable point of these heuristics is that the next generation of individuals is shaped by collecting the best individuals and then integrating them together. Through this integration, the population is enhanced over the succeeding generations. On the basis of this, the optimizer of the exploration stage includes some design parameters that have to be randomized as much as possible to globally explore the promising solution search space. [17][18][32]

  Because of the stochastic-based nature included in the optimization process, picking up the right parameters for an adequate balancing between the exploration-exploitation dilemmas is a serious challenge and perhaps the most critical challenge facing the development stages of any metaheuristic algorithm [33]. The most popular leading examples of this category are Genetic Algorithms (GA), Genetic Programming (GP), Biogeography-Based Optimizer (BBO). [17][18][32][28]

  To sum up, it is vitally important to realize that decision making by using metaheuristics implicit involves a fundamental selection between "Exploration" by which more information is assembled that might direct us to more superior forthcoming decisions or "Exploitation" by which the finest decision is made in the light of the existing knowledge.

- **Swarm Intelligence (SI) algorithms:** These optimization algorithms are evolved out from the collective intelligence and the communication channels that can be observed in the social behavior of the biological populations in nature. They are used to solve most of the optimization problems that arose on the metaheuristics' horizon over recent years. A typical example of this category is the Sea Lion Optimization (SLnO) algorithm that imitates the hunting activities of the sea lions. Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) algorithms are considered as other common examples. [17][18][32]

  Furthermore, these algorithms remain the most fertile research area in the field of metaheuristics. In comparison swarm-based with evolution-based algorithms, the former has some more advantages over the latter. Since evolutionary approaches have relatively more operators than swarm-based, they are more difficult to apply. Different than evolution-based approaches that immediately discard any obtained piece of information related to the old iteration once a new population is generated, swarm-based algorithms normally keep these valuable pieces of information over the subsequent iterations. [17][28][32]

- **Physics-based algorithms:** These algorithms are mainly coined to simulate the physical phenomena in the world. Gravitational Search Algorithm (GSA) is one of the best-known examples of this category. GSA is formulated on both the law of gravity and the law of motion. Harmony Search (HS), and Simulated Annealing (SA) are other dominant examples of this category. [18][28]

- **Chemical-based mechanisms (CBM):** The natural process that involves transforming unstable ingredients into stable ones is named as a chemical reaction. During these interactive operations, excrescent energy exists due to the sequence of elementary interactions between these molecules. But at the end of these transformations, the unstable molecules are converted to stable ones and, naturally, with low energy stability. In this regard, scientists focus their efforts on trying to find algorithms that imitate the chemical interactions among molecules that happen during the chemical reactions and usually lead to chemical changes. Chemical Reaction Optimization (CRO) proposed by Lam and Li (2010) is one of the best-known examples of this category of algorithms. [11][21][34][29]

- **Stochastic optimization (SO) Algorithms:** The formulation of these optimization algorithms includes not only generating random variables to be used in the progress of the searching process but also using methods that have arbitrary (i.e. random) iterate steps. However, the outcome success of the iteration steps couldn't be guaranteed. The followings include broad examples of these algorithms: stochastic hill-climbing, swarm algorithms, evolutionary algorithms, genetic algorithms, simulated annealing, to mention but a few. [29][10]

- **Probabilistic-based Algorithms (PA):** These algorithms are so named because the probabilities play a significant role in making decisions within the different runs (i.e. iteration steps). Simulated Annealing (SA) is mostly the oldest example of this type of algorithms. [10][18][35][32]

- **Artificial Immune Systems (AIS):** As a sub-field of biologically-inspired computing, these artificial intelligence algorithms are mainly concerned with imitating the biological immune processes of the human immune system

towards solving a broad category of different optimization problems from engineering, information technology, and mathematics. [10][28]

- **Artificial neural networks (ANNs):** These algorithms are one of the information processing paradigms and a subfield of biologically-inspired computational intelligence family. Inspired by the manner that biological neural systems process data and based on the principle that self-learning is acquired from experience, these artificial networks need to be trained enough by using a set of examples to create adequate knowledge that can be used later for solving a wide variety of optimization problems in real life. [36][18] [37][29]
- **Human-based algorithms:** Because there are laws governing all the internal operations of the human being, all the contained internal activities of these complicated systems operate functionally without any problems. Attractive by this motivation, scientists from all domains try to simulate the ways in which these subsystems work and come up with new goal-oriented operating methods to solve many important real-world problems. Thus, the algorithms of this category emulate the intelligence and the social behaviors of the human being and their associated activities. Teaching Learning Based Optimization (TLBO) [2][29], Interior Search Algorithm (ISA)[2][29], Colliding Bodies Optimization (CBO)[2][29], and Harmony Search Algorithm (HSA) [30][3][38][29] are broad examples that are classified under this category.

From a broader perspective and under one scheme, these optimization techniques can also be categorized by some wider classifications as the followings:

- They can be categorized as being either exact (i.e. enumerative) or approximated methods [2][10].
- Under another scheme, they can be also categorized according to whether they are used in forming other hybrid metaheuristics or not [29]. Regarding this hybridization, evolutionary and nature-inspired algorithms are the most algorithms that have been extensively hybridized with each other over the last few years to solve a wide variety of optimization problems [10].
- Another further broader categorization can be as conventional metaheuristics, like Genetic Algorithms (GA) which is the most famous and prevalent one, and the new generation ones, such as the proposed algorithm of this paper [29][27]. Compared to the classical ones, these modern algorithms usually require lower computational time and memory, fewer setting parameters to fit the problem, and moreover easier to implement [10][20].
- Another completely different but it is a common classification scheme is the availability or not of local search mechanisms within their stages. Since local searches usually give the best chances for approaching the best candidate solution, this facility gives more feasible chances for the candidate solutions improvement during the course of successive iterations [29][32].

Yet, regardless of the fact that there is a fairly wide range of heuristic and metaheuristic approaches that were proposed so far in the spectrum of the optimization paradigm, there is still immense room for improving and/or investing the available ones or at least coming up with new viable algorithms and techniques like the one described in this paper. This is especially true if the following silent points are under the vision: [13][12][9][4][21][20]

- Most optimization problems that arose on the horizon over recent years are often very hard to be tackled by the conventional models and, hence, they require new ways of thinking to be solved. Anyway, creativity comes from the well recognizing of the problems that require further innovative usages of the optimization algorithms.
- As the scope of the optimization problems is growing extremely in size and heterogeneity, the number of optimization problems residing on diversified domains of our life is exponentially larger than most scientists and researchers have ever proposed.
- Since not all metaheuristics are reported as being successful ones, there remains a relatively substantial research gap that needs to be filled between the small number of accepted metaheuristic methods, from classical to novel approaches, and the vast number of day-to-day optimization problems that are increasingly duplicated.
- Since many of the conventional optimization algorithms used earlier may no longer be sufficient to upkeep the new needs of today's attitudes, it is vitally important to reengineer them or make a permutation for them with new applicable and practical alternatives.
- In the optimization paradigm, it seems so strange and somehow unfamiliar to find a single algorithm that performs well on most optimization problems, especially that a large fraction of them have their own circumstances, requirements, constraints, and implementations scenarios.
- It is regarded as axiomatic that the system that has been constructed to meet the high needs of scalability and reliability has more opportunities to stay functional for a longer time. So, there may be counterintuitive variations in the solution quality between the algorithms that had been implemented and evaluated only on just small or medium benchmark instances of the problem and the algorithms that had been tested on all benchmark instances of various sizes including complex large-scale ones.
- There is a clear and distinguishable variance between both theorizing that is largely based on the theoretical-academic world and the implementation that is conducted upon real-world cases. An analogy with this, there may be some considerable gaps between the metaheuristic theories and their corresponding real-world implementation. This mismatch between both of them is, of course, caused by the

fact that some metaheuristics are relying on just theoretical or abstract possibilities without applying them viably with the real-world applications or that some of them have been tested and then evaluated using only low-to-mid-range data without exposing them hard on large-range data. Close related to this, a considerable fraction of these researches have been originally initiated for only research purposes without being for real-life applications.

More importantly, some metaheuristics are just carried out inside research labs where some of them have been constructed based on hypothetical projections with only an academic or theoretical vision that may be far away from the factual situations. That is, carrying out lab-problems merely without being exposed to diverse real and hard tests is subject to guesswork and experimentation may lead to unexpected results. On top of all that, nearly the majority of these labs are subject to some financial constraints with rare to no external support available. Inevitably, this lack of certainty may rarely lead to unfaithful decisions and hence far-reaching problems.

- Some metaheuristics were conducted upon simulated data that are nearly different from the relevant real-life ones and they may not initially design for fully exploiting the real environment. Since they are firmly governed by purely theoretical standpoints without having a strong empirical base or practical evidence, they could be as a matter of theoretical tests and, for that reason, building knowledge about simulated data could be neither viable nor feasible. This, in a way or another, maybe behind finding some missing environments to conduct experiments on.

Apart from the foregoing mentioned discussion, all metaheuristic optimization approaches are alike on average in terms of their performance. The extensive research studies in this field show that an algorithm may be the topmost choice for some norms of problems, but at the same, it may become to be the inferior selection for other types of problems. On the other hand, since most real-world optimization problems have different needs and requirements that vary from industry to industry, there is no universal algorithm or approach that can be applied to every circumstance, and, therefore, it becomes a challenge to pick up the right algorithm that sufficiently suits these essentials [21][29][12].

What's more, the metaheuristic research community frequently uses the two terms “Heuristics” and “heuristic methods” interchangeably to simply give the same meaning. Like so, the two terms “metaheuristics” and “metaheuristic methods” are also used interchangeably. Furthermore, it is recalled that the following optimization terms will be used to refer to the same metaphor: algorithm, method, and technique.

Table I. Local search heuristic vs metaheuristic strategies

Feature	Heuristics	Metaheuristics
Heuristics level	• Low-level heuristics	• High-level heuristics
Evolution level	• Low	• High
Performance level	• Low	• Generally, they have better performance, but it is not guaranteed.
Domain	• Because they are tailor-made for specific problems, they have a narrower and less generic domain relevant to the problem type. • A large fraction of them is proposed by and for specialists in the same domain. • They are problem-dependent & special-purpose methods. Since they are usually created to solve problems of a particular type (i.e. problem-specific), they are more related to the problem that needs to be solved. Therefore, they usually work poorly when they are applied to solve other problems.	• They are applicable in solving real-life problems that are complex, nonlinear, high dimensional, and multimodal. Moreover, these problems are usually having unknown search space and a massive number of local-optima traps. These problems can be easily seen in many aspects and extents of our day-to-day life, like industry, agriculture, engineering, business, social, and many other fields. • So, these algorithms might be used to solve those problems which are unsolvable. • Since metaheuristics are based on novel and abstract concepts, they allow designing versatile software that can be applied to a broad range of optimization problems covering various domains and disciplines. Hence, they are categorized under the general-purpose methods. • Wider and more generic domain, in relevant to the problem type. • They can be adopted to solve NP-hard problems that can’t be unraveled by utilizing ordinary heuristic methods. • They have multidisciplinary domains and objectives. • Problem-independent & general-purpose heuristics

Feature	Heuristics	Metaheuristics
Searching space	• They have a narrower search criterion. • They are applicable to solve optimization problems whose search landscapes are well-formalized. • They are mostly more exploitative approaches.	• They have a wider search criterion and a broad range of possible scales. So, they are primarily prepared to process instances of large-scale problems. • These algorithms are mainly suitable for solving optimization problems for which the search space isn't well-formalized. • They are mostly more exploratory in nature.
Searching process & Local optima trap	• A local search: The searching process is no more than within the neighborhood surrounding. • They have only one searching rule for guiding the progress of the search process in the search-space. • Thus, they are more prone to stagnation in local optimas. They might get trapped in some local solutions (i.e. local optimums) without making some progress in bypassing them.	• A global and widespread search: Since it is important to look beyond local optimas to find the global optima that any algorithm is ultimately looking for, the searching is relying on the context of generalized searching hoping to be as near as possible to the most optimal solution. • As they have multiple abstract rules and higher-level computational strategies, they are always exploring the search space thoroughly under many highly intelligent scenarios. • They have some mechanisms to evade being stuck in some local optimas traps and in forcing the algorithms to escape away from them. Most of these mechanisms have a stochastic nature to widely search the entire space.
The overall searching behavior	• Single behavior: They have just one and only one behavior in achieving the most optimum solution.	• Multi behaviors: Based on self-adaptive computing, they have multi behaviors that keep changing according to the status of the problem. • They are quite smart to tune their parameters in the direction of finding the most optimum solution with the least possible computational cost.
Abstraction level	• The abstraction level is low and more specific in relevant to the problem type. • They take local views of the considered problems.	• The abstraction level is high and more generic in relevant to the problem type. • Goal-oriented operating methods • Bedside that they are based on abstract concepts, they encompass highly-scalable methodologies. • They take global views for the considered problems. • Complex working details are simply abstracted away from the end-users.
Simplicity	• They are relatively difficult to implement and required more setting parameters to fit the considered problems as it should be. • a far less attractive	• They need fewer control parameters to fit the considered problems. • Since they are based on the usage of simpler concepts and the utilizing of the setting parameters that can be adjusted and tuned to match the problem nature, they are relatively far more attractive and easier to implement.
Reliability and Flexibility	• Capturing every detail of the problem under consideration is always fundamental. • less practical solutions	• Most of them look to the considered problem as a black box that has an easily-known group of input and output as if capturing every low-level detail is not always essential. • Many diverse problems can be solved without much changing in the original algorithm structure.
Other core properties	• Approximated • They require detailed knowledge of the considered problem.	• Approximated, Heuristics & Stochastic & Iterative. • Most of them are nature-inspired. • They can be used in a broad array of problems without any problem-dependent knowledge.

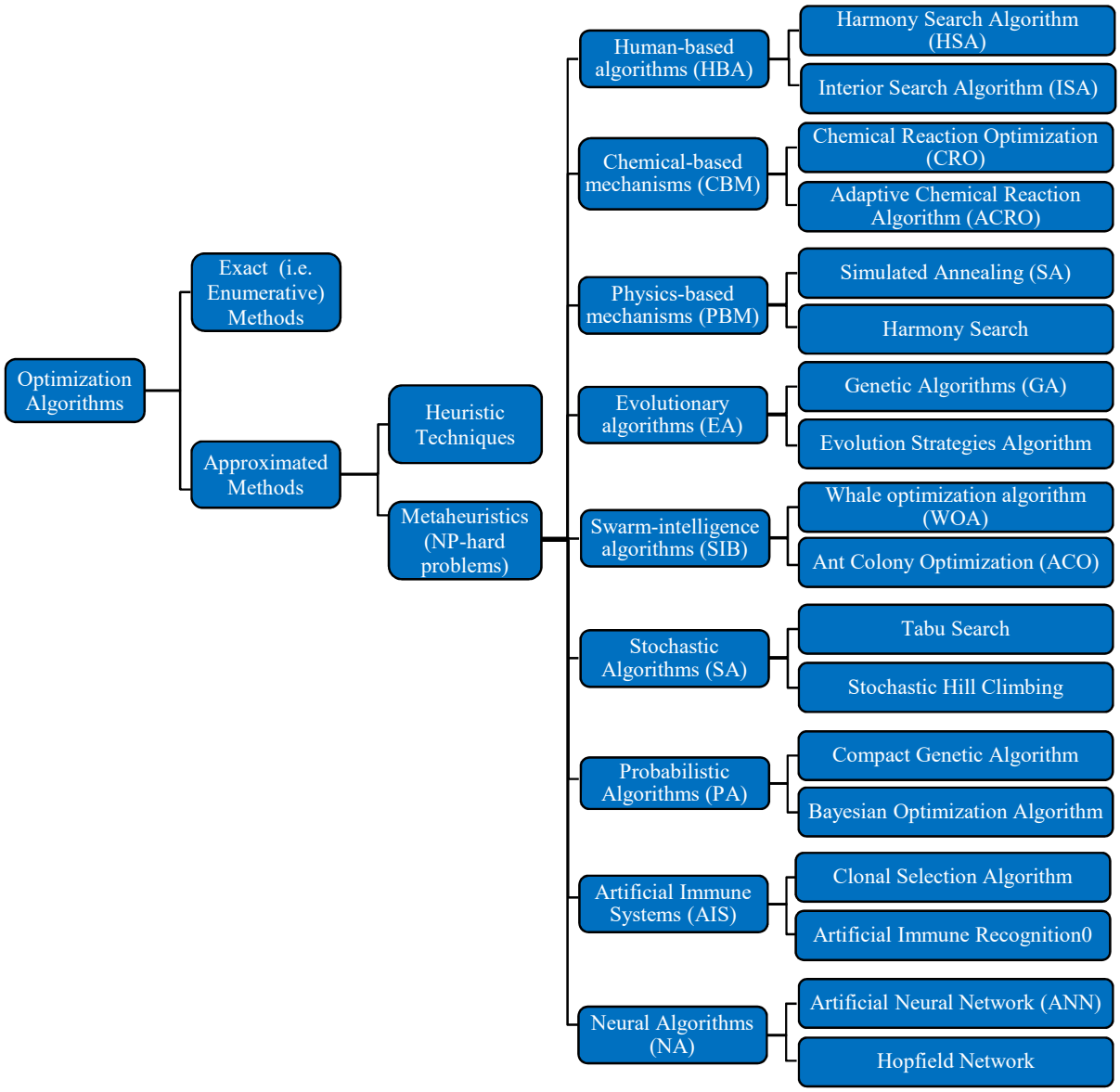

Fig. 5. The various strategies for solving optimization problems and some of their main representatives.

3. Related Work

Related to its importance, researchers, experts, and practitioners all over the globe increasingly extend their literature towards solving the Maximum Flow Problem (MFP) using different methods and techniques. In the course of that, they made every effort in every way they could to propose new potential solutions or modifying the already available ones. Along this way, the first notable solution was presented by Ford and Fulkerson in (1956) by using the augmenting path algorithm which is later known as FF [39][8]. Their algorithm is all about solving a problem that is described as the followings: [39]

"*Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady-state condition, find a maximal flow from one given city to the other.*"

Although the FF algorithm is the most popular one in this paradigm, its overall complexity is comparatively high, which is O(mn). Nonetheless, they remain the founders and the great pioneers of MFP even with that aforesaid complexity, and moreover, their research and standard results remain a benchmark for excellence by which many other researchers compare theirs. [39][8]

Since the seminal paper of FF and throughout the years, researchers and scientists continue their goal-oriented research toward finding various techniques and methods that are revolved around the mission of capturing more optimal solutions for the considered problem. Based on that, the lines of advances have been initiated and a sequence of numerous algorithms was suggested, presented, and released. Because there's a large body of research studies on this line of research and to offer a coherent narrative as an alternative of annotated bibliography, this section is inherently selective to a certain extent, and so there are some of a fair-bit less relevant topics that haven't been presented. Some of these salient studies are, therefore, presented in the following subsections.

3.1 Chemical Reaction Optimization (CRO)

In a broad sense, scientists frequently noted that the nature of both chemical interactions, named as Chemical Reactions (CRs), and optimization paradigms have high-level common attributes and details. At the starting point, the following noteworthy points highlight the common phenomena in between: [13]

- Elements are substances, also better-known as materials, which cannot be reduced to be simpler by the usual chemical means.
- In its simplest terms, the basic unit in any CR is the molecule.
- When a substance is transformed from an unstable case to a more stable one, a chemical change will occur to this substance which is referred to as chemical reaction (CR). In fairness, this chemical process is considered as a natural process and nothing else.
- A collision is the exact cause of any CR. In this regard, the molecules are considered as being the manipulated agents and, therefore, CR is a multi-agent paradigm. Each agent has a number of features, some of which are fundamental to the CR operations. However, other features may be easily attributed to the agent.
- Taking for granted that the CR event is only triggered by the sequence series of collisions in-between molecules (i.e. agents) and nothing else, some rhythmic interactions between these agents may occur that lead to some changes upon these agents themselves. On the other hand, CRs can be commonly categorized into four elementary schemes that are viewed in Table II. Additionally, stability is the primary objective of any CR in which involves changes in the molecules.
- The following triple rules are directly related to energy. First, energy already presents and can't be made. Second, energy may inter-change from one shape to one or more other. Third, energy can't be smashed. Related to the second rule, collisions usually lead to the rearranging of energies among molecules, but there may be a collision in which no energy is transferred.
- Both CRs and optimization undergo a sequence of step-by-step events. They both strive in finding the optimal solution or at least the near-optimal one.

Table II. The four major types of chemical reactions

Name	General Reaction Pattern	A chemical formula example
Combination or synthesis reactions	A + B ⇨ AB	$S + O_2 \Rightarrow SO_2$
Decomposition reactions	AB ⇨ A + B	$CaCO_3 \Rightarrow CaO + CO_2$
Substitution or single replacement reactions	A + BC ⇨ B + AC	$H_2 + 2\ AgNO_3 \Rightarrow 2\ Ag + 2\ HNO_3$
Metathesis or double displacement reactions	AB + CD ⇨ AD + CB	HCl + NaOH ⇨ NaCl + HOH

Based on manipulating the above-mentioned observations, especially the step-wise process of searching, many researchers aim to relate the chemical reactions with the optimization paradigm and, as a result, try to embed all the common concepts and properties between them in new optimization algorithms. Consequently, they proposed many general-purpose metaheuristic potential algorithms that emulate by the natural process of chemical reactions. Then, they successfully utilized these algorithms with the intention of resolving a broad range of both discrete and continuous engineering problems which cannot be underestimated. In all cases, it is important to take into account that these chemical-reaction-inspired algorithms are often population-based and have a high ability to be adapted to cover other problems. They are commonly referred to as Chemical Reaction Optimization (CRO) algorithms. [13]

For satisfiability, the core of the overall CRO-based algorithms is primarily all about the followings [13]:

- Compared with the other classical algorithms, CRO offers some flexibility to be customized and controlled by the users themselves to fine suit their specific needs or to be easily adapted to address particular problems.
- In order to reach or at least approach the global optima, CRO-based algorithms are a self-adapted to reflect the problem domain.
- In reality, CRO has the ability to solve some optimization problems which have not earlier been successfully tackled

by other metaheuristic algorithms or that have been classified as having some run-time complicity issues.

- Including C++ and Java, CROs can be easily coded using object-oriented programming (OOP) languages. In this context, the molecules are defined as classes and the elementary reaction types are defined by the methods.
- For solving a particular problem, multiple CROs can be implemented simultaneously without any trouble.
- Since each CRO keeps up its particular relevant population size, it will not have to remain pending at any certain instant until any other CRO accomplishing its certain tasks.

To be truthful, the CRO algorithm was originally devised up by Lam and Li (2010) for the purpose of fixing the combinatorial optimization problems [21]. Just within less than two years, CRO was applied successfully to resolve a considerable number of optimization problems, outperforming several other existing algorithms in the majority of the experimental results [13]. Through the said research, they put the generic formulation for any optimization problem. In terms of this, they mathematically define the minimization objective function "f" by utilizing Equation 1: [21]

$$\min_{X \epsilon R^n} f(x) \text{ subject to } \begin{cases} C_{i(X)}=0 & i \epsilon E \\ C_{i(X)} \leq 0 & i \epsilon I \end{cases} \quad (1)$$

Where the following points analyze the elements of this equation:

- "R", "E", and "I" represent the real number set, the index set for equalities, and the index set for inequalities, respectively.
- $X = \{x_1, x_2, x_3, \dots, x_n\}$ and $C = \{c_1, c_2, c_3, \dots, c_n\}$ are the vectors of variables and constraints, respectively. "n" and "m" are the problem dimension and the total number of constraints, respectively.
- If a negative sign is added to "f", then it will be the maximization objective function.

In order to evaluate the performance of the proposed solution, the simulation code has been implemented using the Microsoft Visual C++ programming language. The algorithm has been applied successfully in solving 23 large-scale instances in which their considered datasets were categorized as being NP-hard of the type Quadratic Assignment Problem (QAP). They observed that the proposed algorithm achieved the objective drastically. Then, an ample computational investigation was carried out in which the simulation results of the proposed algorithm were evaluated and compared to the best performing three metaheuristics recommended in the literature at that time, relating to the solutions' quality level and their computational execution times. These three competitors are: Fast Ant System (FANT), an Improved Annealing Scheme (ISA), and Robust Taboo Search procedure (TABU). Moreover and to offer more objective among the other three compared metaheuristics and to eliminate any issues related to the variations in the execution environment, the same implementation environment was used related to the computer type and model, operating system, the function evaluation limit of the stopping criterion, and all the other standard measures. In most of the cases, their pilot experiments gained the best result and that is why this proposed algorithm is among the current best algorithms which can be used to solve QAP. Because it can be used as a generic searching algorithm to formulate several NP-hard problems, their algorithm is part of importance and remarkableness.[21]

After the antecedent algorithm successfully solves a variety of optimization problems, Lam and Victor (2012) presented another expanded research for solving a wide variety of engineering problems, such as the quadratic assignment problem, multimodal continuous problems, ANN training, and other optimization problems. During their notable research, they build a roadmap framework and theoretical guidelines recommending other users on how to customize and tune the CRO's setting parameters to match the nature of the other problems. Besides that, their effective research is considered as a tutorial and practical procedure that encourages other researchers in exploiting CRO in solving their research problems. In other words, their research study is an inspiration for every optimization research which comes along. [13]

The study by Barham et al. (2016) introduced another noteworthy CRO algorithm which is conducted using JAVA programming language. This CRO achieves an overall complexity of "$O(I\ E^2)$", where "I" and "E" indicate the number of iterations and arcs of the directed-weighted graph, respectively. They prove that the number of iterations has assured evidence towards capturing additional optimal solutions and approaching the most optimal ones. [34]

3.1 Whale Optimization Algorithm (WOA)

On the relatively species-rich sea, humpback whales need a developed strategy in their hunting for together. These whales types actively hunt small fish or krill, following them according to tight enough coherent strategy. This foraging social process for self-maintaining is a unique interaction that hasn't been detected in other creatures yet. It is interesting to note that this type of social creature has no teeth and above that, it has a very narrow throat and so, this is the rationale behind that it couldn't swallow large prey as a whole. However, this type of whales has an amazing policy in attacking a great group of small prey and catching them, the studies find. This unique to the concept foraging policy is called "bubble-net feeding" and it is based upon a multi-stage coordinated mechanism for capturing as much as possible fish at once. Once they are teaming up together, they dive brilliantly beneath a large group of prey and then all begin cleverly in bubbling out to produce a net made of bubbles and forcing prey to be inside. To make sure that the net of bubbles surrounds all the prey, they should reinforce all the net's weak points and, accordingly, they splash their flippers (i.e. fins) at these weak parts. By this witty tactic, a large group of prey is trapped tightly inside a well-organized

fence isolated from outside, and so the only remaining event to do is swallowing all of them by the helping of their flippers that swiftly direct fish headed for their mouths. [26][40][29]

Whale Optimization Algorithm (WOA) is proposed by Mirjalili and Lewis (2016) [26] which is considered as a new competitive swarm-based optimization algorithm that evolved mainly out of abstracting the fascinating hunting behavior of the humpback whales. With this article, the researchers have successfully created a mathematical model to match the humpback whales' feeding strategy upon which many NP-Hard optimization problems have been solved. This mathematical model begins initially with a population of various stochastic solutions, each of which is generated by a search agent (i.e. a humpback whale). Whenever the best solution is determined among the other ones, all the other search agents should arrange their current locations accordingly. On the other hand, this research also addresses the case where these animals may make a random search moving towards finding other better positions instead of remaining stuck with one of the current search solutions. [26][29]

In order to test and evaluate the algorithm, WOA was implemented empirically by solving 35 real-life optimization problems of practical importance, 29 of them are mathematical and the remainders are structural design. Furthermore, WOA was verified to evaluate its performance using classical benchmark functions that are usually utilized in the optimization literature. Each one of the considered experiments was iterated thirty times. After WOA is compared with other conventional techniques, the gaining results were relatively competitive. Due to its considerable success, this algorithm becomes popular from then on. Day by day, a sequence of similar research was conducted based on WOA. [26]

Any electric power system has been considered to entail three functional zones. First of all, the electricity generation by which the energy is transformed from the available resources into electric power. Secondly, the transmission of bulk electric power over long distances by using high-voltage networks. Thirdly, the distribution which is related to providing the end consumers' low-voltage service points from the high-voltage networks. On the condition that the energy is consumed directly by those end consumers as soon as it is changed into electric power in the first functional stage, the important point related to the entire system is that there is an electrical power loss and the largest portion of this is routinely occurring at the distribution level; it is about 70% of the total circuit loss. Due to this, increasing the overall energy efficiency of any distribution path is the hardest part of setting up any electric power system. The study by Reddy et al. (2107) is based on WOA by clearly decreasing the generating plants' losing power during this distribution. In the long run, this study can be used not only in reducing the voltage and the high power loss but also in lowering the cost and producing stable, efficient voltages by optimizing the placement and sizing the distributed generators (DGs). [38]

Back-and-forth, Masadeh et al. (2018) suggested the "MaxFlow-WOA" algorithm that is based mainly on the Whale Optimization Algorithm (WOA). The proposed solution was compared to Ford-Fulkerson's MF with respect to the accuracy of the results and the average computational run-time where it acquired "$O(E^2)$" as the overall time complexity. According to the authors' experimental analysis, the impressive experimental results give sufficient sound evidence and reinforce the conclusion that "MaxFlow-WOA" is an effective metaheuristic for solving the Maximum Flow Problem (MFP). [1]

Combining the WOA and the rapid and the big advances in the distributed parallel applications of metaheuristics, a parallel whale optimization (Parallel-MaxFlow-WOA) algorithm is developed by Masadeh et al. (2020) to solve the MFP. But the truth, this algorithm is considered as a more powerful and an expanded version of the sequential MaxFlow-WOA. It works by segmenting the search space (i.e. the network graph) into four segments, all of which are computed in conjunction with each other. Then, the best maximum flow of these segments is selected. The algorithm was tested on different datasets that have between 50 to 1000 vertices and the number of edges between 502498 to 50024998. Then, the algorithm's solution quality was evaluated for each dataset. Compared to the FF sequential algorithm, the proposed algorithm achieved a tangible (3.79) reduction in the overall computational running time by running the segments on four-independent-parallel processors. As this result is a great enhancement of the computing time, the first noticeable impression of this four-part segmentation stimulates the authors to strongly recommend applying this proposed algorithm using the distributed systems architectures, at least to gain their parallelism powerful benefits. Table III shows a computational complexity comparison between the FF, Sequential-MaxFlow-WOA, and Parallel-MaxFlow-WOA. [40]

Table III. Complexity comparison between FF, Sequential-MaxFlow-WOA, and Parallel-MaxFlow-WOA

Complexity type	FF	Sequential MaxFlow-WOA	Parallel MaxFlow-WOA	Note
Augmenting path cost	O (mV)	O (|E|)	O (|E|)	• "**m**" denotes the number of the arcs. • "**N**" denotes the number of clusters. • "**V**" is the number of humpback whales (i.e. vertices or nodes). • "**E**" is the number of edges in the directed-weighted flow graph.
Run-time complexity	O ($\|V\| + \|E\|^2$)	O ($\|V\| + \|E\|^2$)	O ($\|V\| + \|E\|^2$)	
The overall computational running time	O ($\|V\| + \|E\|^2$)	N * O ($\|V\| + \|E\|^2$)	Max (O ($\|V\| + \|E\|^2$)$_N$)	
The maximum flow	|E|	|E|	|E|	

3.2 Grey Wolf Optimization (GWO)

Within the animal kingdom, an animal itself may be the predator and/or the eaten prey of the others. Grey wolves as predators are mostly known to prey on a variety of large, hoofed animals such as bison, mountain goats, moose, and other different kinds of deer. These wolves, also formerly known as "Canis Lupus", are living in packs with an average size between five and twelve; they have a very strict leadership hierarchy or perhaps not surprisingly, one of the most fascinating social behaviors one has ever seen. Mirjalili et al. (2014) tried to mathematically simulate this dominance hierarchy structure and the sovereignty levels during the social hunting practice and, for that, a going-on-down-hierarchy algorithm is designed that is closely related to this. This algorithm is called Grey Wolf Optimization (GWO). As illustrated in both Fig. 6 and Table IV, the dominance level in this suggested tactic reclines from the top towards the bottom and can be carried out at four-hierarchical commanding levels: Alpha which is the dominant leader the one with the topmost liability, Beta which is Alpha's assistance for decision-making, Delta which controls Omega, and Omega which are under the domination of the other wolves. Granted, each team member is classified as being one of these four levels. [16]

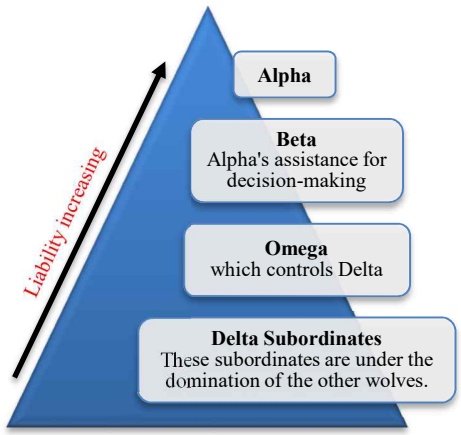

Fig. 6. Pyramid of the leadership hierarchy for the grey wolves

Since catching and killing the prey should be simulated with a particular interest, a three-stage algorithm that matches up to their hunting plan of action is proposed in this research paper. First, the exploration stage when these predators are searching the area (i.e. the search space) in pursuit of prey. Second, what they are doing when they are finding certain prey attain their request; they pursue and then try to harass it to encircle. This stage ends when the prey is encircled and stops moving to escape. Third, the exploitation stage which represents the scenario where the actual attack process will be launched on. At this stage, Alpha begins a sheer attack on the prey. Whenever Alpha needs assistance, Beta and Delta help in the attacking process, but under the control of alpha. In short, every wolf is part of a one-team attack that organized and carried out the execution. [16]

Table IV. Grey wolves' dominance structure

Name	General Reaction Pattern	Duties	Solution Hierarchy
Alpha	The dominant leader, i.e. the one with the highest liability.	Design the hunting plan, the place to sleep, the time to sleep or wake up, and so on.	The best solution
Beta	The deputy leader and the commander alternative.	The Alpha's decision-making assistance that rules the other lower-level members. Besides it is the pack's educator, it supplies alpha with any constructive feedbacks rendered to carry out and support the Liability. Since it is the commander alternative, it will be appointed as a leader in the case of Alpha's dysfunctionality.	The second best candidate solution (i.e. the second level in the hierarchy).
Delta	They lead omega, i.e. members of omega are dominated by delta.	The general care and safeguard responsibilities are attached to them. Hunters, scouts, experts, ex-alphas, ex-betas, and caretakers are belonging to this category.	The third best candidate solutions
Omega	The working class which is under the domination of the other wolves (i.e. Subordinates).		The underneath level in the hierarchy.

To ensure that the algorithm' efficacy standard is of a high level, the following investigations were applied in [16]:

- To benchmark its performance, the GWO algorithm was evaluated against a comparison group containing twenty-nine prominent test functions which were categorized as benchmark tests. The algorithm was executed and iterated thirty times on every one of these benchmark functions.
- The quality of the proposed algorithm was also compared with the other five well-known metaheuristics recommended in the literature which are Differential Evolution (DE), Evolution Strategy (ES), Particle Swarm Optimization (PSO), Evolutionary Programming (EP), and Gravitational Search Algorithm (GSA).
- To further properly teste and extensively investigate its behavior, all the aforesaid benchmark functions are

employed in testing the algorithm in terms of the followings:

- **Exploration:** in relevant to this point, the algorithm provides either a merit result when it has provided very competitive outcomes compare to FEP and DE or a distinction grade when it has outperformed GSA and PSO, as well.
- **Exploitation:** the algorithm provided outstanding performance and presented highly competitive outcomes in the matter of exploiting the optimum solution with the least possible computational cost.
- **Local optima avoidance:** At this point, the algorithm extremely has outperforms the others in at least half of the benchmark functions and has also provided successful competitive results in the second half.
- **Convergence:** the behavior of the algorithm proves that it has the ability to eventually convergences at a point in the search space.

To tell the whole truth, the ability of the algorithm in making and controlling a fair balancing between the two cornerstones of any metaheuristic algorithm, exploration and exploitation, has resulted in successfully going beyond most local-optima traps.

- An important step forward was to proceed in evaluating the algorithm on a wide range of three large and difficult instances of engineering design problems that are quite popular among researchers around the world. These challenging problems are tension/compression spring, welded beam, and pressure vessel designs. The algorithm indicated a high-performance capability in solving these problems.
- The said authors also took into account proving the performance of the algorithm in an authentic application and, in turn, the algorithm was inspected in an optical engineering problem which is referred to as optical buffer design. This problem is directly related to one of the internal key elements of the optical CPUs. All over again, the algorithm is also able to solve this real-world application and can be widely adopted by the optical CPU industry

To put it in a nutshell, all the above-mentioned experimental analyses and the comparisons that have been made with the other approaches have guaranteed that the GWO algorithm is able to offer more competitive results and it is applicable in many challenging problems, especially those that have unknown search spaces.

Since then, the pyramid of the grey wolves' commanding levels has garnered a lot of attention from the researchers and, so, another GWO-based research has been held by Masadeh et al. (2017) for solving the maximum flow problem (MFP), named MaxFlow-GWO. The authors utilized the K-means technique to divide the grey wolves into groups, called clusters, each has its leadership and encloses by five-to-twelve wolfs. Accordingly, the graph is segmented into clusters and each five-to-twelve vertices are grouped together as one cluster. Then, they proposed a three-stage algorithm that matched up with their corresponding fishing scenario: searching for prey, encircling prey, and attacking prey. The time-growth complexity of this proposed algorithm is "O(|n| + |E|2)", where "***E***" indicates the number of arcs (i.e. the number of edges between wolves) and "***n***" stands to the number of vertices (i.e. wolves). After the computational run time of this algorithm was compared with the well-known Ford-Fulkerson' algorithm by using the same datasets, the achieved outstanding results were significantly more optimal. [24]

3.3 Metaheuristics' Parallelism

As the systems of the clustered parallel data processing can be employed for producing high-quality results and, at the same time, surpassing the calculation speed, the artificial intelligence (AI) specialists and other interrelated participants exploited this distributed architecture in carrying out their High-Performance Computing (HPC) codes to process large-scale problems. To address this, the same network graph is divided into a number of subgraphs based on the existing number of processors. Each subgraph contains a number of augmenting paths. Then the calculation of the whole network graph is distributed among a number of processors where each subgraph is computed alone by a single processor. On the grounds of this, all the processors cooperate together to solve the problem simultaneously. [11] [12][41]

In this context, some Jordanian researchers use IMAN1 supercomputer which is located in Jordan to conduct their experiments. It was assembled using 2260 Sony PlayStation3 (PS3) devices that are linked together via a fiber-based network. Combined, this supercomputer provides multiple resources, high-end integrated clusters, and an open parallel and distributed computing environment. Besides that this supercomputer has an extraordinary efficiency such as its capability in driving 25 trillion operations per second and serving thousands of concurrent clients with sub-millisecond latency, it has also many supporting powerful tools meeting both industrial and academic computing needs performing millions of simultaneous input/output actions. [42]

With the intention of solving the MFP by utilizing the modern technology of IMAN1, a parallel genetic algorithm (PGA) is suggested by Surakhi et al. (2017) which is an extension to the serial version of the algorithm [41]. Based on a real distributed system, they exploited the HPC cluster architecture in designing the stages for each one of the iterations to work simultaneously in conjunction with the others. After the network's directed-weighted graph is segmented into a set of subgraphs, all the different augmenting routes from the start node "***s***" going to the target node "***t***" are altogether computed concurrently through the using of the so-popular message passing interface (MPI) library which is a standard library used for multi-core multi-thread parallel execution development. As each subgraph has its own local Maximum Flow (MF) solution,

all the local MFs contribute together in generating the overall global MFP solution. And so, the total maximum flow value for every one of the iterations can be produced by the simple summation of these augmenting routes. Consequently, this value is subject to be enhanced from iteration to another one. Compared with the sequential version of this algorithm, the needed time was rationed by 50% and they have achieved more worthy results in terms of accuracy, speed, and time efficiency.

In their ways of seeking high-level computation with respect to the quality of the solutions and the response time, IMAN1 as a supercomputing center is used yet again by Alkhanafseh et al. (2017) in solving some of the chemical reactions problems. As an improvement of the standard Chemical Reaction Optimization (CRO) algorithm, the authors proposed a modified version of the serial CRO algorithm where the main problem graph was segmented into a set of subgraphs distributed on multiple processors; each one was responsible for computing one augmenting path. In comparison with the sequential CRO, good enhancement was significantly achieved by applying the parallel version in terms of the quality of solutions and the overall computational running times. The time needed (i.e. time complexity) to solve the maximum flow instance for the parallel implementation of the algorithm is "***O(NEF * P)***" where "***N***", "***E***", "***F***", and "***P***" are the count of vertices (i.e. nodes) for each subgraph of the flow network, the count of edges for each augmenting path between the source and the sink vertices, the maximum flow value from the source vertex to the sink vertex, and the number of concurrent processors that are used in execution the algorithm, respectively. [11]

Going with the new vision of the smart digital world, El-Omari within two research articles, (2019) and (2020), ends up that the CC digital-priceless environment is truly the most tolerable place for hosting many complicated algorithms especially that it has actually emanated out from unlimited and never-dying diverse resources for ever-sooner and inexpensive calculation and, moreover, it has the world's best price-per-performance ratios for the HPC operations. The author also pointed out that there are genuinely thousands to millions of virtual machines (VMs) that are dynamically generated and demolished to serve numerous customers in easily ending their very sophisticated or even unmanageable tasks. Accordingly, the author pointed, at least implicitly, to another further step forward by building the needed optimization algorithms remotely as web-based innovative services within the CC environments. From over here, it is foreseen in the next few coming years that global solving for the optimization problems based on utilizing CC might become a wildly-popular simple practice among ordinary users. [12][43]

3.4 Artificial Neural Network (ANN)

ANNs are a family of computational intelligent models that strive to simulate the process of exchanging messages among neural networks' biological systems that especially exist inside the human and the animals' brains. In analogy to the brains' working process when catching information, these artificial nervous models are used particularly to solve the optimization problems that have an extreme number of inputs in which most of them are usually unknown. The neurons are imitated in these artificial models as nodes connected with each other to form an artificial model viewed as a network of nodes. The important point is that every connection connects two nodes is associated with a given numerical value that represents its weight, called the neuron's activation value. These connection weights are determined by feeding the training data set to the input layer and adjusting these weights recursively over the course of the training process' iterations. Since the acquired knowledge is learned accumulatively from the collected data, these neural networks need to be trained sufficiently on a fairly large set of data that is relevant to the problem domain. During the training process or as a so-called learning process, these weights are fine-tuned based on the extracted knowledge; this step is regarded as the most crucial one to accomplish high recognition accuracy. From a purely practical standpoint, the training process should be repeated until the network is capable of learning and adaptive to the different inputs. [36][18]

The objective of the learning process is to analyze, summarize, extract, and elicit the associated knowledge from the training data set. All these foregoing errands entail a deep understanding of the basic structures of the training data set. Even though exploring great amounts of data with the purpose of retrieving some relevant knowledge could be a frustrating and complicated task, the acquired knowledge could be latter used successfully in detecting patterns and trends for the sake of classifying information.

Since ANNs were introduced in the pattern recognition field and optimizing nonlinear functions that work recursively, El-Omari (2008) developed a new robust technique that intends to optimize the segmentation of the compound images based on modeling the solution sample space using the ANN paradigm. By building prior knowledge utilizing four interconnected ANN's as a one-model component, an image can be segmented by this brilliant approach into labeled and coherent regions of four classes: pictures, graphics, texts, and backgrounds. Then each region is manipulated individually according to its characteristics with the most effective compression method that either a new one or one off-the-shelf. In that research work, there were another three proposed techniques to meet the preceding stated segmentation objective; all of them revolve around a vivid central point by modeling the ANN but each one has its own layered-structure topology and characteristics related to the applied evaluation criteria (i.e. activation function), the number of the hidden layers and the number of nodes (i.e. neurons) in each one of the hidden layers.

A fifth hybrid approach is further proposed in that research work as a result of hybridizing these four stated approaches. However, each of these proposed approaches has it is certain trade-offs such as speed, reliability, accuracy, efficiency, and ease of use. In fairness, the main concern of these proposed

models, particularly the last approach, is related to the high-level computational power and the time complicity where the author of the said study suggested two highly efficient strategies to this end. The first one is by building the segmentation methodology as utility software that may be considered as part of any operating system. The second strategy is by building the data segmentation mechanism internally as a special-purpose built-in chip. Within these two strategies, every image is right away encoded when it is stored and, vice versa, decoded when it is retrieved back. [18][43]

Another relevant ANN research was developed by El-Omari et al. (2012) for optimizing the segmentation process of the compound images by using both the multilayer feed-forward Artificial Neural Network (FF-ANN) and the Back-Propagation (BP) learning methodology to feed-backward the losses. BP, also called the error-BP method, is so named because of the difference (i.e. error) between the desired outcome of the current neural network iteration step, "O_d", and the preceding one, "O_a" is fed back to the same neural network, namely from the output units to the input ones. The mathematical construction of this main difference can be modeled as in Equation 2: [36]

$$\mathbf{E = 0.5 * (}O_d \mathbf{ - } O_a\mathbf{)^2} \quad (2)$$

In the course of this added-value work, the proposed algorithm breaks down any compound image into equal-size-square blocks. The network was designed and modeled to discover and identify patterns that are relevant to the set of the various components of the block. Since the right training of the neural network is the most critical side of building a reliable model, the neural network was modeled and implemented using MATLAB® and trained empirically upon a collected database containing 2987 24-bit-RGB-bitmap images of different resolutions, each has its own features. After the necessary proper-sufficient training of this neural network to ensure lower error rates, the neural network weights were tuned properly many times and then each block was evaluated to determine its type and then classified accordingly. Once the blocks are been classified as it's intended to, all adjacent blocks having alike features and classes are fused together to form a single and coherent whole region. From the observation of actual practices, the nuanced outcomes indicate that the proposed model is successfully capable to define the types of each block with an average accuracy of 89% and broadly applicable. [36]

Despite its low error rates, the noteworthy issue of the proposed solution of [36] is that the training process of ANN, especially over big data, is seemed somewhat computationally expensive and depends on a large number of hard-headed setting parameters that demand extra steering effort to fit the considered problem. So there is a necessity to enhance the overall performance by speeding up the ANN's learning activity. To make things most sense, substituting the training functions of the learning process by more-advanced metaheuristic algorithms may accelerate the ANN model and, in turn, better performance may be achieved by solving this drawback.

Since, utilizing the advantage of metaheuristic in hybridization ANN with another metaheuristic may solve the abovementioned performance weakness of this algorithm, the first author of that study (i.e. [36]) is currently in the process of implementing this proposed algorithm, namely SLnO-MFP, as an alternative to the Back-Propagation (BP) function. By this inclusion and integration, the suggested algorithm works as an effective tool within the ANN's training. This may have a strong probability in raising the outcomes' efficiency and in reducing the maximum number of generations that are necessary for the elicitation process of the final solution which as an axiomatic in-line with reducing the time needed to implement them. Furthermore, this may theoretically make the probability of the optimal solution arrival even greater. For a more detailed explanation and illustration of this algorithm, the interested reader can refer to the mentioned paper.

3.5 Artificial Bee Colony (ABC)

The differentiation between honeybees' behavior and computer also attracted hundreds of researchers in proposing some artificial intelligence algorithms used to solve many real-life problems. The researchers found these bees live in groups called colonies where each bee colony, also referred to as hive, has at least three well-known subgroups of bees: scout bees that responsible for searching for the new food sources (i.e. solutions) which are the flower nectar, onlooker bees which knew the amounts and determine the exact places of any food source by watching the dancing ways of the scout bees, and the employed bees which are responsible for gathering the food from the resources' places that are defined by the scouts. They also found the members of each group (i.e. colony), as well as the subgroups, have their own structure for the working tasks and dominance hierarchy. [31][29]

By studying the behaviors of these colonies especially how all the bees contribute together in generating the optimal solution of the nectar harvest, the research work held by Saab et al. (2009) introduced a novel and valuable optimization algorithm based on using the Artificial Bee Colony (ABC) optimization. With the condition that the probability of choosing any candidate solutions (i.e. flower nectar as the food source) is directly connected with the fitness function (i.e. nectar's amount, nectar's quality, and the distance between the colony and the food's source), the importance of their algorithm in the real-world is its ability to balance between the two searching phases exploration and exploitation in the searching iteration steps around finding and reaping the flower nectar. For a more detailed explanation and illustration of this algorithm, the interested reader can refer to the mentioned paper. According to the real implementations of the two scenarios of scouting and forging processes, this algorithm can be used to employ many real-life optimization problems that don't demand supervision which includes, but are not limited to, the following examples: combinatorial optimization problems, stochastic problems, multi-targets, data-mining-search-engine

crawling, parallel implementation, multi-targets, and parallel implementations. [31]

3.6 Hyper-heuristic Framework

A number of researches have been made in an effort to find solutions related to the optimization of the execution time issues. In this regard, the study by Welch and Miller (2014) offered a two-level model for optimization algorithms turnaround. The primary premise of this much-appreciated model is to separate the functionalities of the metaheuristic optimization algorithm from the functionalities of the problem itself. Broadly, this is the notion of the hyper-heuristic framework where a set of intelligent metaheuristics can be classified upon their shared common features into different types of hyper-heuristics and then combined together through the hybridization of the hyper-heuristics framework. Then, rather than exploring the search space of the candidate possible solutions for a given problem, the framework of the hyper-heuristics automatically fabricates an algorithm that could professionally find a better solution by one or more of the already metaheuristics that have been classified and stored. Besides that this hybridization leads to new approaches to emerge, this combining of the positive capabilities of the different metaheuristics gives more chances for capturing better solutions. While two or more approaches that participated in this hybridization can be fused together to form one model, each one of them remains functioning as an individual one. [5][29]

From a broader standpoint, hyper-heuristic frameworks can be understood in such a way as if there were two abstracted parts associated with each other and maintaining a high degree of correlations: a top-level frontend and a lower-level backend. While the metaheuristics themselves are encapsulated to form the backend, the frontend which involves different types of hyper-heuristics is the visible part that the optimization handler sees and interacts with. By this easy-to-implement way of using the hyper-heuristic frameworks, a higher level of abstraction is provided where the backend complexity is shielded from the frontend and so some unwanted details are eliminated or hidden. This architecture allows those handlers their selves to focus their effort on solving the problems rather than concentrating on the minutiae of the underlying details related to how the metaheuristics should be implemented for solving the considered problem. In all sincerity, hyper-heuristic does not revolutionize the field of metaheuristics, but it adds a new easier and quicker face in dealing with them.[5]

3.7 The new-generation metaheuristics

Finally, to conclude the discussion of this section, the study by Dokeroglu et al. (2019) introduced a considerable selective survey to compare the most popular metaheuristic algorithms that have been proposed in the last twenty years, namely between the years 2000 and 2020. This distinguishable survey reviewed and then analyzed the new-generation metaheuristics in such a way that it can be considered as an excellent benchmark for the metaheuristics comparison in the optimization area. On part of comparative performance measurement in that 20-year span, this prominent study drew an objective comparison between the metaheuristics according to the following five critical issues: [29]

- How many setting parameters are required to go efficiently with the optimization task of the compared algorithm? Since setting every input parameter requests more time and effort to be adjusted and tuned to go with the problem nature, the algorithm with the lesser parameters is, without a doubt, the better choice to use. From a deeper viewpoint, the number of parameters in any metaheuristic is directly proportional to its complexity where the algorithm with fewer parameters needs slighter variations to work well in solving other problems and, as a result, to dominate over a larger variety of applications.
- Which are the stages of the metaheuristic algorithm that have the ability to balance properly between the exploration and exploitation strategies? As a consequence of the metaheuristics' stochastic nature, optimal balancing between these two stages becomes unquestionably a challenge to meet.
- Has the metaheuristic been used in developing other hybridized approaches? There is no question that the algorithm that is used much in other hybridized approaches is credible evidence of its well-built organized structure efficiency.
- Does the metaheuristic algorithm contain some local-search mechanisms? Since local searches usually give the best chances to keep approaching the best solutions, the presence of this facility has a considerable influence on the candidate solutions improvement during the course of the successive iterations.
- Does the metaheuristic algorithm search the solution space globally for catching the optimal solution or just within the local solution space?

Granted, any metaheuristic algorithm satisfying the above features will have more stability to be staying used in the upcoming years. From another point of view, this analytical study could highlight some clues as to how to screen and select the most adequate metaheuristic for a given optimization problem. [29]

Furthermore, this research study drew attention to two other important issues that may prevent some metaheuristics from being utilized. One is related to the lack of a solid analytical foundation for validating many metaheuristics in which their performance evaluations are measured only by carrying out the classical ad-hoc statistical analysis without being based on real theoretical or mathematical foundations. Then the said study also pointed out how it is difficult to find clear guidance or robust frameworks to recommend anyone interested in how to adapt many metaheuristics to the considered problems. [29]

Related to the previously mentioned setting parameters issue debated in [29], the authors in the study by Lam and Li (2010) emphasized that the adjustments of many metaheuristics are just relying on the experience and prioritize of the researchers themselves without applying clear conceptual structures and theoretical guidelines for how to set and tune their input parameters according to a problem domain [21].

4. Maximum Flow Problem (MFP)

Since the efficient flow supplying should be guaranteed by a well-defined distribution system, MFP is one of the pillars of computer engineering, mathematics, and computer science that has been deeply studied [22]. From a conceptual standpoint, MFP as a research area is indeed classified as being based on the fusion of three-well-known-overlapped branches that are directly interrelated with both Artificial Intelligence (AI) and Operations Research (OR). First, Computational Complexity Analysis (CCA) which is pertaining to the cost of solving computational problems [22]. Second, Nature Inspired Computing (NIC) which is an emerging research area that mainly concentrates its focus on solving various global optimization problems based on exploiting efficiently Chemistry, Physics, and Biology approaches [3][20]. Third, Combinatorial Optimization Problem (COP) that tries to make every great effort into achieving an "optimal" solution among a set of possible candidate ones [34].

Before describing how the proposed algorithm solves the problem under consideration, it may be well to first consider the matters of some notations and fundamental features. First of all, any type of these problems (i.e. MFPs) considerably includes the following common components:

- $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots, \boldsymbol{x_n}\}$: This combination of variables is formally referred to as the set of the initialization parameters and so the problem dimension is represented by "$\boldsymbol{n}$". It also indicates the vector of variables that are used to frame the designed methodology of the proposed algorithm. For its time convergence and to explore a much broader diversity inside the candidate solutions, the impressive achieved results of any algorithm are highly interrelated with the choice of this set.
- $\boldsymbol{C} = \{\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \dots, \boldsymbol{c_m}\}$ represents the vector of constraints (i.e. criterions or rules) that are used to restrict the progress of the searching process. From another perspective, this vector is used for the purpose of limiting the various values that are assigned to the vector "$\boldsymbol{X}$". On the assumption that the total number of constraints is defined by "$\boldsymbol{m}$", all these constraints are so sacred not to be violated while discovering the optimized solution. It is vitally important for any solution to have complied with this vector of constraints. Prompted by this, a feasible solution is considered as a potential one if it certainly satisfies all the indicated constraints along.
- "$\boldsymbol{S}$" represents the solution space. This vector represents the set of all possible candidate solutions for a given problem. [5][4]
- "$\boldsymbol{s}$" indicates the set of values that are assigned to the vector "$\boldsymbol{X}$" and restricted by the vector of constraints "$\boldsymbol{C}$".
- "$\boldsymbol{F}$" represents the objective function (i.e. objective criterion) that is used to assess the quality level of a stated solution. It represents the criteria used to pick out the optimal solution among the possible candidate ones. [4][7]

Second, with the aim of using SLnO in solving the problem of interest, the input of the maximum flow problem (MFP) should be adapted into a layout format that can be understood by SLnO. Thus, the input should be converted into a graph. Broadly and from a mathematical modeling part, this can be understood in an abstracted aspect by using the following bullet points that represent the problem instance:

- The shape of the flow network is called "network connectivity". This network is a directed graph, "$\boldsymbol{G}$" that has a finite set of directed weighted arcs (i.e. edges), "$\boldsymbol{E}$", and a non-empty finite set of vertices (i.e. nodes), "$\boldsymbol{V}$". That is to say, the MFP can be defined as "$\boldsymbol{G = (V, E)}$" where the following points clarify this: [40][7][8][44]
  - "$\boldsymbol{G}$" indicates the weighted directed network graph, called a digraph.
  - "$\boldsymbol{V}$" denotes the group of nodes, also named vertices, which are inside "$\boldsymbol{G}$"; their count is "**n**". In an analogy with Fig. 1, $\boldsymbol{V} = \{\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{t}\}$ and **n=7**.
  - In order to carry flow, nodes are combined together by the set of arcs "$\boldsymbol{E}$", also referred to as edges. The behavior of the network "$\boldsymbol{G}$" is defined by the way that the set of nodes "$\boldsymbol{V}$" are connected via this set "$\boldsymbol{E}$" and by the strength of these connections, called weights or capacities (i.e. flow). All edges' weights are assumed to have strictly positive values and, conventionally, these weights are set to be small numbers [29][2].
  - A variable for each edge of the graph is introduced in the graph of MFPs. For instance, the edge between the two nodes "$\boldsymbol{a}$" and "$\boldsymbol{b}$" of Fig. 1 is represented by using the variable "$\boldsymbol{e_{ab}}$" where "$\boldsymbol{e_{ab}} \in \boldsymbol{E}$".
  - The representation of the weights inside the set "$\boldsymbol{E}$" is used to represent the search agents. Every edge "$\boldsymbol{e_{ij}} \in \boldsymbol{E}$" that is directed from a given node "$\boldsymbol{i}$" towards another node "$\boldsymbol{j}$" has a maximum of non-negative capacity "$\boldsymbol{c_{ij}}$". The total number of edges inside the network "$\boldsymbol{G}$" is "$\boldsymbol{m}$". By looking at Fig. 1, the total number of edges is 10, hence **m=10**.
- Two special nodes in "$\boldsymbol{G}$" are distinguishable and designated in advance as follows: [44]
  - A source or a start node "$\boldsymbol{s}$" in which flow is arriving. Unlike the other nodes, the node distributes flow to the other nodes. It has only an incoming flow without outgoing flow.

- A sink or a target node "t" out which flow is leaving. It has only an outgoing flow without incoming flow.

- Owing to the fact that "G" is a directed graph without having any self-loop (i.e. an arc connecting the same vertex). Strictly speaking, "i" and "j" should be different vertices in any arc "$e_{ij} \in E$". Obviously, "e_{ij}" and "e_{ji}" refer to two distinct edges.

- "E" is represented equivalently by a symmetric adjacency matrix "E" of "V x V" in size, also known as the truth-assignment matrix. This is depicted in Fig. 7:

E =

	s	a	b	c	d	e	t
s	0	1	0	1	0	1	0
a	0	0	1	0	0	0	0
b	0	1	0	0	0	1	1
c	0	0	0	0	1	0	0
d	0	0	0	0	0	0	1
e	0	0	0	1	0	0	0
t	0	0	0	0	0	0	0

Fig. 7. The adjacency matrix "**E**"

Where "$e_{ij} \in E$" is equal to one if there is an edge that connects the two vertices "i" and "j", and zero otherwise. It is worth noting in this matrix that "t" has no edges emanates from it which justifies why that row contains just zero values. Similar to this, "t" have no edges emanates into it, and so its column is of zero values. Note also that the two diagonals both have zero values since no self-loop.

- Since this network graph is a directed graph and all edges are associated with given and declared weights, this type of network is referred to as a "labeled weighted graph".

- Related to this proposed algorithm, the node "t" in the network "G" refers to the prey (i.e. sink) and the nodes (i.e. vertices) represent the sea lions.

- The capacity constraint: Suppose that both "i" and "j" are two nodes in "V", then: [7][8]
  - The directed arc "$e_{ij} \in E$" interconnects this pair of nodes with respect to the direction from "i" to "j".
  - Every directed arc "$e_{ij} \in E$" has an associated non-negative integer flow capacity that is denoted by "u_{ij}". This capacity represents the maximum amount of flow that can be transferred from the node "i" to the node "j" through the arc "e_{ij}" measured by the capacity unit. Taking into account that "$u_{ij} \in U$" also represents the cost of fitness between the two vertices "i" and "j" of "G". By referring to Fig. 1, the weighted edge "e_{ba}", for instance, has a capacity of fourteen units (i.e. e_{ba}=**14**) while "e_{ab}" has a capacity of fifteen units (i.e. e_{ba}=**15**). Certainly, all of these weights are crucial for the construction of this network and the flow on every arc should not exceed its capacity.
  - In synonymous with Fig. 1, the weights' matrix "U" is represented equivalently by a symmetric adjacency matrix "U" of "V x V" in size. This matrix is associated with every weighted directed network "G". Hence, the association of these weights to all edges of the network "G" of Fig. 1 is shown in the matrix of Fig. 8:

U =

	s	a	b	c	d	e	t
s	00	08	00	25	00	25	00
a	00	00	15	00	00	00	00
b	00	14	00	00	00	06	12
c	00	00	00	00	13	00	00
d	00	00	00	00	00	00	29
e	00	00	00	17	00	00	00
t	00	00	00	00	00	00	00

Fig. 8. The maximum capacity matrix "**U**" for every pair of vertices

In this matrix, if there is an edge between the pair of vertices "i" and "j", then "U_{ij}" is equal to the capacity of that edge, and zero otherwise. It should also be noted that "U_{ij}" is different than "U_{ji}".

- Since a network flow is only as large as its slowest edge, the trick of investing the most out of the network "G" is to brilliantly make sure that each edge "$e_{ij} \in E$" is giving of its best and then eradicate any possible bottlenecks or any potential conflicts between their parts. In synonymous with Fig. 1 and Fig. 8, the flow' matrix "F" is represented equivalently by a symmetric adjacency matrix "F" of "V x V" in size. This matrix is again associated with every weighted directed network "G". Hence, the association of these weights for every pair of vertices of this network "G" is shown in Fig. 9:

F =

	s	a	b	c	d	e	t
s	00	05	00	04	00	06	00
a	00	00	10	00	00	00	00
b	00	09	00	00	00	03	04
c	00	00	00	00	04	00	00
d	00	00	00	00	00	00	11
e	00	00	00	10	00	00	00
t	00	00	00	00	00	00	00

Fig. 9. The actual flow matrix "**F**"

Where this matrix has to fulfill the following two conditions:

- If there is an edge between the two vertices "i" and "j", then "$f_{ij} \in F$" is equal to the flow of that edge, and zero otherwise.
- The flow of any edge "$e_{ij} \in E$" couldn't exceed the upper-bound capacity value of that edge (i.e. $f_{ij} \leq u_{ij}$) for every "$f_{ij} \in F$" and "$u_{ij} \in E$" [44].

- Related to the network graph, it is easy to see that Equation 2 holds: [44]

$$\sum_{(s,u)\in E} f_{su} = \sum_{(u,v)\in E} f_{uv} \quad (2)$$

That is the flow value of a network "***G***" is the sum of all flows that get formed in the source node, "***s***", or equivalently of the flows that are used up in the sink node, "***t***" [44]. By looking at Fig. 1, for instance, it can be seen that both the incoming flow at the target node "***t***" and the leaving flow at the starting node, "***s***" are equal to fifteen.

- Augmenting path: The set of all the arcs emanating from node "***i***" is denoted by "***E(i)***". From another standpoint, "***E(i)***" refers to the all edges that are connected with "***i***". In relevant to this, a track (i.e. a path or a route) with available capacity is referred to as an augmenting path.
- The conservation constraint: Assume that there are many routes joining between the beginning node, "***s***", and the corresponding destination sink node, "***t***", the maximum flow capacity is defined as the maximum total value of flow that can be moved where satisfying the following constraints is definitely a must by any possible candidate solution [7][8]:
  - Every arc "$e_{ij} \in E$" must satisfy all the mass balanced constraints along.
  - Since the graph of interest can be represented and used as a basis to generate the flow, the total arriving flow and the total leaving flow for every node "***i***", other than "***s***" and "***t***", should be absolutely equal to each other. This can be mathematically expressed as it can be shown in Equation 3:

$$U = max\{u_{ij} \ by \ (i,j) \ in \ E\} \quad (3)$$

In other words, for every other node "***i***", other than the source or a start node "***s***" and the sink or a target node "***t***", both values of the incoming (i.e. arriving) flow and the departing (i.e. leaving) flow must be equal. Otherwise, the network "***G***" couldn't accurately map the input flow into the output flow which means that some given capacities values, "u_{ij}" are possibly missed or incorrect.

  - The total entire flow leaving "***s***" is equal to the total incoming flow arriving in "***t***". The value of the flow is defined by using the formulas described in Equation 4 and Equation 5:

$$|f| = \sum_{v\in V} f(s,v) = \sum_{v\in V} f(v,t) \quad (4)$$

That is,

$$Max\, f(x) = \sum_{(u,v)\in E(S)}^{n} X_{uv} \quad (5)$$

Where it is easy to see that the followings hold:

$$\sum_{\{v:(u,v)\in E\}} X_{UV} - \sum_{\{V:(V,U)\in E} X_{vu} = 0; For\ all\ \ u \in V\{s,t\} \quad (6)$$

$$0 \le X_{uv} \le F_u \ For\ all\ \ u \in E \quad (7)$$

$$\vec{D} = \left| \vec{C} \ . \ \vec{X}_P(t) - \vec{x}(t) \right| \quad (8)$$

5. Sea Lion Optimization (SLnO) Algorithm

The term flock or flocking can be used to refer in particular to the group of birds. On the other hand, swarm behavior or swarming, as a term, is originally applied to insects but it can also be used interchangeably to refer to any other collection of interacting creatures or entities that in their normal lives aggregate together in swarms and interact harmoniously with each other according to two norms of interaction that are mainly governed by laws of nature. The first norm is related to the local interaction between each other, and the second one is their collective interaction with their environment in which joint hunting is the most distinguishable one among these core activities. This collective behavior or as called collective intelligence or global behavior, of these groups of creatures has recently gained significant attention from scientists and researchers from different research institutes and universities around the world and, for that, a new science called swarm intelligence based (SIB) has been coined as shared knowledge and collective concepts of a sequence of algorithms and models. On the grounds of this, this science is based mainly on the natural social behavior of the biological populations. [26][45][31]

Motivated by the above-stated inspirations, scientists try to come up with new goal-oriented techniques to tackle various real-world optimization problems. And, thus, the so-called "nature-inspired computing" is evolved out as a field of computer science that is targeted directly toward making computers imitate the intelligence of the swarms (i.e. SIB). Broadly speaking, solving optimization problems, particularly the complex ones, and operating numerous complicated systems can be drawn out by this field of science, namely SI. [13][18]

In view of this, many metaheuristic algorithms have been proposed that take direct inspiration by the collective intelligence and the ways of exchanging in-depth information among each other [31]. These algorithms are used expansively not only in resolving optimization problems but also with many other real-world automation fields like healthcare, manufacturing, military, and other related domains [3].

The sea lions are amphibians and aggregate inside swarms called colonies where each colony has a massive number of members that are grouped into subgroups. While the whole group has a leader, each subgroup has a leader, as well. The important point is that each colony can be considered as a hierarchical paradigm where all the low-level components (i.e. subgroups) inside this hierarchy work together to form a higher level-hierarch. The whole system behavior is determined by aggregating and integrating together all the lower components of this hierarchy to form a higher collective behavior or as so named global behavior. Based on the commanding orders, the joined members of any subgroup can be moved to another

subgroup inside this hierarchy according to their age, gender, and the tasks that are entrusted to them. However, each member may be subject to this moving over and over through their lifetimes. The whole group or some of its subgroups may hunt together which increases their chances of gaining more feeding prey. From this, a self-organized teamwork system is formed by grouping together and coordinating all the different interactions within the lower level of the hierarchy.

Compared to the other creatures feeding on the relatively species-rich seawater, each colony has a very strict coherent strategy in their fishing together. In the subject of the optimization literature, Sea Lion Optimization (SLnO) algorithm is relatively one of the best-known dominant examples of this category which is based mainly on imitating both the hunting manners and associated direct communication channels among any swarm of sea lions during their chase for the preys [2][26].

There is a continued need for all the swarm members to interact with each other by direct communication channels upon seeing prey. Chasing and catching the detected prey is the joint venture of all the related swarm members. In accordance with their social hunting nature, moving towards an optimal target implicitly exists in their primitiveness and all the associated activities are parts of their nature. Since it is a joint hunting process, it is clear that the primary responsibility for hunting implementation and success rests with all the agents themselves.

As do all the other nature-inspired paradigms, SLnO is a population-based algorithm that is based on using the fitness concepts to determine the quality of the solution. By this means, the proposed algorithm starts by generating a population containing multiple search agents. To assess these search agents, the fitness function of an efficient solution which is based on the maximum value is computed for every search agent. At that instant, the best search agent "$\overrightarrow{SL}$" who has the best fitness among the group of candidate solutions is defined and all the related locations belonging to the other search agents should be modified directly in-keeping-with this modification.

Out of all of this, there is a pressing need to address the following appealing concerns related to the organizational structure of their social behavior: [46][2][31]

- **Solution search space:** What are their feasible regions, namely their solution search space? How are sea lions responding to any vital action in the hope of finding more superior solutions? Is it an on-demand and self-adaptive strategy? How are these animals moving in the direction of the optimum goal or near-optimum one?
- **Hunting process:** Do the sea lions have a strategy for feeding or it is a trivial-usual process and a sudden inspiration? If there is a hunting strategy, is it a hunting-for-together strategy? How does the hunting process work? Is there a to-do list of basic tasks that should be initiated? Is this foraging process behavioral, evolutionary, or both? Is the hunting process a one-team concerted effort?
- **Collective behavior:** Is the collective behavior of the sea lions centralized or decentralized? In other words, does the collective behavior of the sea lions rely on only one sea lion that is responsible for making every single decision related to the whole swarm, and so it is a centralized system, or several sea lions are responsible for making decisions, and so it is a decentralized system or as so-called self-organized.
- **Communication process:** How are sea lions communicating with each other? What are the guidelines for managing this communication process in the followed direction? How do they communicate to update their locations to be turned towards the new target position of the prey after any movement?

Along the way of revealing these research concerns, the SIB, in particular the sea lions which are considered the major inspiration of this technique, is systemically the collective behavior of the following attractive notions and patterns of behaviors: [46][2][13][31]

- **Solution search space:** The theater of the events depends on the environment. It is the sea beach in the case of sea lions. The whole graph is the search space (i.e. search agents) and the prey is the target node that these animals are looking to reach and catch.
- **Collective knowledge:** Given that the teamwork of the sea lions is greatly self-organized, global behavior is essentially derived from and based on the authority of self-organized agents and their tendency in reaching the goal. Therefore, the emergence of collective behavior is achieved by collecting all the local communications between all individuals. This leads to having a coin with two faces: the formation of the tuned-global-collective knowledge arises from the exchange of the different information among the sea lions and, on the other side, the different rhythmic interactions and communications among individuals in the system lead to global-collective coordination. And so, the depth of collective knowledge is based on this teamwork communication.

  It is worth mentioning that a successful solution is constructed by a subscription of all agents and a single sea lion, on his own, has a lower possibility to efficiently solve a problem. On the other hand, it is an undeniable fact that the collection of the sea lions composing the team has an overall stronger possibility of getting closer productive results and better markedly solutions.
- **Natural laws:** Behind the scenes, functions and operations of the wholly-embedded components are regulated according to natural laws. For this well-defined reason, these high-level systems activate repetitively forever without any troubles.

- **Hunting stages:** To track and hunt the prey in case of the sea lions, the chronological framework for the hunting events goes through five stages:
  - Detecting and tracking phase: The sea lions have faces with elliptical cross-sections that are different than the other mammals which have faces with circular cross-sections. In addition to that, they have the longest whiskers among all the mammals which can be moved in all directions. Depending on the feeling of these whiskers, these animals track and determine precisely all the related information concerning the prey such as location, shape, and size. [2][47]

    Furthermore, sea lions habitually swimming randomly in a zigzag course during their searching for prey across the sea. As well, they utilize their whiskers to get a keen sense of the prey. [47]
  - Searching for prey (Exploration phase): When the prey is composed of few fish, sea lions hunt individually. Otherwise, when the prey is composed of plenty of fish, they are chasing down together and hunting together as groups. The sea lion (search agent) who successfully detects the position of the prey is considered as the best search agent and, in turn, this lion is assigned as the leader that commands the hunting process. This leader starts the process of hunting by telling and guiding the other members about the prey which is considered as the current best candidate solution [2][47]. Whenever the best search agent is determined as a leader among the other ones, all the other search agents should arrange and update their current locations accordingly.

    Nevertheless, if a better prey is detected by another search agent, then this new prey is considered as the new best candidate solution. In view of that new situation, the leader, as well as the current best candidate solution is replaced.
  - Vocalization phase: many sea lions from a variety of swarms begin to group together (i.e. forming a cluster) around the prey and so the cooperative clusters are formulated. The key significant factor of this stage is how fast their immediate reactions to the prey movement as soon as the prey position is determined. On the basis of that, the sea lions are chasing down together to force the prey headed for narrow balls at the shallow water near the ocean's surface and the beach.
  - Attacking phase (Exploitation phase): the encircling process which is related to the process of getting around the target prey after determining its position. This process is directed by the leader of the sea lions and requires updating the search agents' positions according to these new circumstances.
  - The actual feeding process: When the prey becomes close to the surface of the ocean, the feeding process is started.

- **The distance function:** The best candidate solution for the sea lions is represented by the current best location that has the minimum distance from the target prey which the swarm has obtained yet. All the joined members should keep track of this location and update their locations accordingly. Equation 9 is used here to mathematically model this behavior which is the most significant characteristic of this technique [2]:

$$\overrightarrow{Dst} = abs\left(\overrightarrow{2B}.\overrightarrow{P(t)} - \overrightarrow{SL(t)}\right) \quad (9)$$

  Where "$\overrightarrow{Dst}$", "$\vec{B}$", "t", "$\overrightarrow{P(t)}$", and "$\overrightarrow{SL(t)}$" represent the distance vector between the sea lion and the target prey, a random vector in [0, 1], the current iteration, the position vector of the target prey, and the position vector of the sea lion, respectively. It is important to draw attention that the vector "$\vec{B}$" is duplicated when it is multiplied by the number two in order to give the search space a closer opportunity to explore a more optimal solution.

- **The positions vectors:** The vectors of the sea lions' positions in any subsequent iteration "$t + 1$" are depending on the preceding iteration which is denoted by "t". This is mathematically modeled by using Equation 10 [2]:

$$\overrightarrow{SL(t+1)} = \overrightarrow{P(t)} - \overrightarrow{Dst}.\vec{C} \quad (10)$$

  Where the vector "$\boldsymbol{P(t)}$" points to the position of the target prey and the vector "$\overrightarrow{Dst}$" is as it has already indicated in Equation 9. The vector "$\vec{C}$" in this equation is reduced linearly from "**2**" to "**0**" throughout the expanse of iteration steps for the reason that this reduction drives the leader of the sea lions into moving in the direction of the current target prey and encircle them. Conversely, an increase in this vector means that the sea lion leader is moving away from the current prey. Thus, the aptitude of the current position of the leader leads to the following three cases:
  - If the value of "$\vec{C}$" is less than one, then the search agent is moving in the direction of the prey and the other search agents should adjust their locations according to that.
  - If the value of "$\vec{C}$" is greater than one, then the search agent is moving away from the prey.
  - If the value of "$\vec{C}$" is equal to zero, then the optimal solution has attained which means that the algorithm terminates at this point.

  However, if the value of ($|\vec{C}|$) is greater than one or less than a negative one, the search agents will move obliquely away and search for a new cluster to join it.

- **The Shrinking encircling mechanism:** This mechanism relies basically on utilizing Equation 10 that has already been mentioned in the previous point.

- **The collective communication:** Since there is a need for the sea lions (i.e. agents) to contact each other and to bring all of them closer together particularly when they are

tracing and hunting as subgroups, much of the success of the SLnO algorithm lies in the collective intelligence that is based on the collective communication. The more collective communication and interaction between agents' population, the more effective collective intelligence and, in turn, the system will be more efficient in solving the problems.

The sound as the communication language is formed by using several vocalizations. Despite the smallness of their ears compared to their bodies, sea lions have got the ability to clearly detect both the sounds that are in the air as well as that are underwater. Sea lions use this communication behavior to call up other joined members who are currently presenting on the beach to join the team immediately and to manage the different hunting activities like tracking and encircling prey. In this regard, Equation 11 is fabricated as [2]:

$$\overrightarrow{SP}_{leader} = \mathbf{abs}\left(\frac{\overrightarrow{sn\ \alpha}\left(1+\overrightarrow{sn\ \beta}\right)}{\overrightarrow{sn\ \beta}}\right) \quad (11)$$

Where the three vectors "$\overrightarrow{SP}_{leader}$ " , "$\overrightarrow{sn\ \alpha}$", and "$\overrightarrow{sn\ \beta}$" represent the speed of sound of the leader of the sea lions, the speed of sound in the water medium, and the speed of sound in the air medium, respectively.

Normally, sound travels faster in solids than liquids and slower in gases than liquids [48]. Under normal conditions, sound travels in water nearly 4.3 times as fast as in air [48]. Otherwise speaking, the sound of the leader needs to be reflected in the two mediums: water and air that are determined by "$\overrightarrow{sn\ \alpha}$" and "$\overrightarrow{sn\ \beta}$", respectively. This sound reflection of the leader of these animals is behind calling the other joined members that are inside the water or at the sea beach.

- **The circular updating of positions:** this behavior is mathematically formulated by using Equation 12:

$$\overrightarrow{\mathbf{SL}}(\mathbf{t}+\mathbf{1}) = \mathbf{abs}\left(\overrightarrow{\mathbf{P}}(\mathbf{t}) - \overrightarrow{\mathbf{SL}}(\mathbf{t})\right).\mathbf{cos}\,(\mathbf{2\pi m}) + \overrightarrow{\mathbf{P}}(\mathbf{t}) \quad (12)$$

Taking into account the fact that the target fishes are the best optimal solution, the following points analyze the elements of this equation:

  - The term "$\boldsymbol{abs}\left(\overrightarrow{\boldsymbol{P}}(\boldsymbol{t}) - \overrightarrow{\boldsymbol{SL}}(\boldsymbol{t})\right)$" represents the absolute distance value between the search agent which is the sea lion and the best optimal solution which is the target prey.
  - "$\boldsymbol{m}$" is a real random number between "**-1**" and "**+1**".
  - The term "$\boldsymbol{cos}\,(\boldsymbol{2\pi m})$" is mathematically expressed to indicate that the sea lion (i.e. the search agent) starts the eating process by swallowing the target fishes that are existing at the bait ball (i.e. prey) edges. Thus, it moves in a circular shape around the best optimal solution (i.e. target prey).

- **The global optimizer:** In order to solve the MFP problem, the proposed SLnO algorithm involves two activates: exploration and exploitation. In the exploitation phase, the joined members modify their locations in light of the best search agent's position. In the exploration phase, on the other hand, the locations of the joined members (search agents) are updated in accordance with the position of the selected sea lion that has been chosen randomly. Therefore, the generalized mathematical formulation of this phase is formulated by using both Equation 13 and Equation 14:

$$\overrightarrow{Dist} = \boldsymbol{abs}\left(\overrightarrow{2B}.\overrightarrow{SL}_{rnd}(t) - \overrightarrow{SL(t)}\right) \quad (13)$$

$$\overrightarrow{SL}(t+1) = \overrightarrow{SL}_{rnd}(t) - \overrightarrow{Dist}.\vec{C} \quad (14)$$

Where "$\overrightarrow{SL}_{rnd}(t)$" is used here to point to a sea lion that is selected randomly from the present population. It should be stressed that when the vector "$\vec{C}$" is bigger than one, this equation is used for detecting the global optimal solution. Because of that, this algorithm is considered as being a global optimizer.

At the early phase of the iteration steps, Equation 14 demands sea lions to randomly proceed around each other. On the other hand, Equation 12 permits other sea lions to reposition themselves or move in a circular shape in the direction of the best search agent which draws the reason behind that this proposed algorithm has high exploitation. In addition to this high exploitation, this algorithm has also a high exploration and the capability to go beyond local-optima traps.

- Related to the graph theory, the sea lions are represented by agents and, in turn, this algorithm is a multi-agent algorithm. The followings are beyond this point:
  - The maximum flow problem is considered as one of the various well known basic problems of optimization in weighted directed graphs. It is a type of network optimization problem in the flow graph theory.
  - The SLnO graph-based for sea lions is usually bidirectional.
  - The weight at each edge (arc) interconnects two vertices (nodes) representing the flow capacity of this arc.
  - The inputs should be converted to a graph with nodes and weighted edges
- All the operations and data-processing activities are ordinarily goal-oriented and real-time functions.
- All the team members (i.e. agents) are accelerating toward discovering better solutions. Much, if not all, of the success of the team, seems to lay upon the tendency of all team members to hurtle past their target.
- Since the supervision of SIB is a self-organized natural system, it is a decentralized system which means that making decisions at the different hunting levels is rooted in the team-environment not only in their leaders. In other words, the fine-tuned vision comes from the fact that all the

sea lions at every level are having some aspect of autonomy in fabricating the working decisions which means in a way or another that all the sea lions are participating seriously in making joint decisions towards achieving the joint goals.

- The promising area(s) of the solution search space, i.e. the feasible region(s), is shared by all the possible swarm members.
- The accumulated knowledge arises over time and, eventually, a closely inter-related vision could be achieved among the swarms' groups inside the same environment. Using the interaction rules, the collected knowledge of the environment may be dynamically updated on the bases of the last acquired information received from the most successful agents that relatively have the closest distance to the prey. This, in the long run, reinforces the subject matter of having joint knowledge. Overall, all individuals of the same population will have the same last updated version of the accumulated information.
- In the case of the sea lions, all the individuals in the same population should adapt their positions and schemata in line with the recent version of joint knowledge that has been built.
- At each iteration step (i.e. epoch) and before the termination of the searching process, the algorithm may search the surrounding habitat for global marked solutions that were not at their local searches. It is often the case that the sea lions typically search according to the position of prey. But, in some instances, these animals may diverge away from each other in looking for the prey and then converge in other instances to attack the prey as one team. As such, these compulsive improvers may be behind finding the global optimal solution.

6.SLnO-MFP algorithm

In this research paper, Sea Lion Optimization (SLnO) algorithm is employed as a means to solve the maximum flow (MF) problem. To achieve this, the same network graph is divided into a set of subgraphs. Each subgraph contains a number of augmenting paths and it has its own local Maximum Flow (MF) which is the highest computed flow for that cluster, called maximal flow. This flow is locally the optimal flow calculated for that cluster and it is different than the global maximum flow computed for the whole graph.

On the other hand, all the local MFs of the subgraphs contribute together in generating the overall global MFP solution, called maximum flow. And so, the total maximum flow assignment value for every one of the iterations can be produced by the simple summation of these augmenting routes.

To address this, the proposed technique is theoretically analyzed, implemented, and evaluated on various datasets. As illustrated in Fig. 10, the core of the SLnO-MFP algorithm consists of four sequenced stages that are fairly complementary to each other; they are detailed in the following subsections.

Description	This algorithm is a population-based metaheuristic. It is important to emphasize that all the local MFs contribute together in generating the overall single-global MFP solution.
	Method
1.	// Initialize the search agents
2.	Generate a population of "n" sea lions $\{\overrightarrow{SL_1}, \overrightarrow{SL_2}, \dots, \overrightarrow{SL_i}, \dots, \overrightarrow{SL_n}\}$ where (**i**= 1, 2 … **n**) and the parameter "n" indicates the population size.
3.	Choose the two special nodes: the source node "s" (i.e. sea lion) and the destination target node "t" (i.e. prey) randomly.
4.	Compute the distance between each search **agent (j)** and "$\overrightarrow{SL_{rnd}}$" by utilizing Equation 9.
5.	**If sea lion (i)** does not join any **cluster**
6.	Join the **sea lion (i)** to the nearest "$\overrightarrow{SL_{rnd}}$".
7.	**End If**
8.	Compute the **fitness function** for each **sea lion (i)**. // The quality of the position of the **ith** sea lion is determined here.
9.	// Call the cluster to which the sea lion will belong to.
10.	Call the **cluster function**. // See Fig. 13 for the detail of this function.
11.	Global **MF** = the Summation of all the Local **MF**s
12.	**Return** the best candidate solution ($\overrightarrow{SL}$) that has been observed. // This is the output of the algorithm.

Fig. 10. SLnO-MFP algorithm

6.1 Initialization Stage

The main algorithm that contains the population initialization of the search agents (sea lions) is depicted in Fig. 10. In this algorithm, a bunch of initial candidate solutions is generated randomly. Overall, this set is so-named as a swarm. Next, two of these search agents are selected randomly, one to refer to the source and the other to refer to the destination which is actually the prey itself. Then, the fitness function for every search agent of the population is computed. For the sake of that, the distance between each sea lion and "$\overrightarrow{SL}_{rnd}$" is computed. According to the nearest "$\overrightarrow{SL}_{rnd}$", the sea lion will be assigned to the nearest group (i.e. cluster).

On the other hand, the issue of enhancement to find the best solution for this norm of optimization problems is considered the most worthy of this algorithm. In order to avoid falling in the local optima and to converge to the best solution within the predefined time, this algorithm also addresses the case where these lions may make a random search moving towards finding other better positions instead of remaining stuck with one of the current search solutions (i.e. one of the local optimas); this what is known as "exploration". In this feature, as soon as the best solution has been determined among the other ones, all the other search agents should rearrange their current locations accordingly.

6.2 Fitness Function

The goodness of the overall solution is evaluated thoroughly by the quality of each possible position which is determined by using the fitness function. The perfect choice of the fitness function has, therefore, a great impact on the selection of the candidate solutions and in the direct evaluation process for identifying the solutions' qualities based on the degree of efficiency. Based on that, there is a need to re-compute the fitness function for each one of the search agents in any new trial. Thus, the best search agent "$\overrightarrow{SL}$" is chosen and all the locations of the other search agents should be updated accordingly.

By using its special vocalization to tell them about the prey, "$\overrightarrow{SL}_{rnd}$", as a leader, send a vocal message to other sea lions that exist on the shore or under the water. In accordance with that, all the sea lions that have heard the vocalization of their leader will join the cluster and then update their locations toward the "$\overrightarrow{SL}_{rnd}$" position depending on the value of ($|\vec{C}|$). Hence, the general steps for updating the positions of these animals are clearly depicted in both of the figures Fig. 11 and Fig. 12.

6.3 Clustering

The philosophy behind clustering is the decomposing of the original flow problem into a number of tractable subproblems (i.e. local MFs). Then, a range of near-optimal solutions to each one of these smaller subproblems is calculated. After that, the collection of solutions for these subproblems is combined altogether to create a global solution, namely global MF.

After the initialization stage and determining the fitness function, the proposed algorithm can proceed forward to the clustering of the network graph. This clustering is used for the purpose of finding the overall solution for a given network graph where the global search space (global MF) is broken down into a set of local search spaces (local MFs), each is referred to as a cluster. Each cluster contains a number of separated subnetworks and each subnetwork is composed of a group of nodes and their edges.

For satisfactory, "$\overrightarrow{SL}_{rnd}$" is selected randomly for each cluster. Then the fitness function for each search agent is computed to check whether it should join any cluster or not. More precisely, each particular agent (sea lion) is managed in the sense that it is identified to which cluster it belongs. So, according to the value of the fitness function, each sea lion is identified whether it will join this group or another.

In order to get the overall global maximum flow ($\mathbf{maxFbw}^{\mathbf{gbbal}}$), the local maximum flow ($\boldsymbol{maxFbw}_{\boldsymbol{i}}^{\boldsymbol{bcal}}$) is computed for each specific cluster and then the overall summation of them is computed. This is illustrated in Fig. 13.

6.4 Maximum Flow Function

As mentioned in the preceding subsection, the local MFP is calculated for each cluster by calling MFP function which is introduced by Ford Fulkerson (FF). MFP function relies on augmentation paths in residual graphs to find the maximum flow from the source to the destination (i.e. source-to-sink path).

The local MFP is computed for each cluster using the FF technique that returns the local MFP for each specific one. Then, the global Maximum Flow ($\boldsymbol{MF}^{\boldsymbol{gbbal}}$) of the network is calculated using Equation 15:

$$\text{maxFlow}^{\text{gbbal}} = \sum_{i=1}^{N}(\text{maxFlow}_{i}^{\text{bcal}}) \quad (15)$$

Where "$\boldsymbol{maxFbw}_{\boldsymbol{i}}^{\boldsymbol{bcal}}$", and "$\boldsymbol{N}$" represent the maximum flow for $\mathbf{i^{th}}$ cluster and the count of the clusters, respectively. On this point, the algorithm shown in Fig. 14 is used to calculate the Local Maximum Flow (i.e. $\boldsymbol{maxFbw}^{\text{bcal}}$) for each cluster.

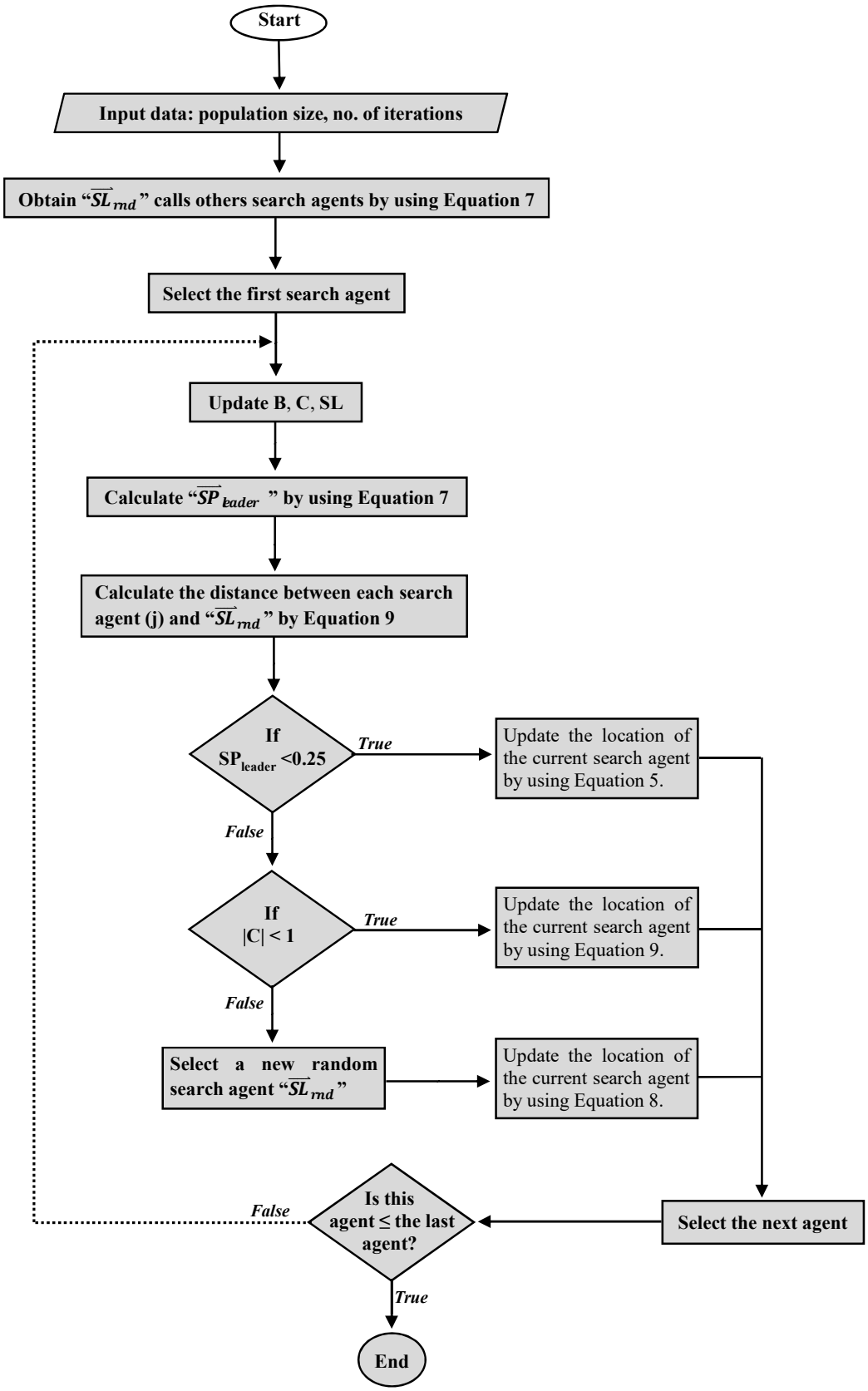

Fig. 11. The flowchart of the proposed framework

Description	This algorithm is all about updating the locations of the sea lions. "$\vec{B}$" is a random vector in [0, 1] and "$\vec{C}$" is reduced linearly from 2 to 0 throughout the course of iteration steps.
	Method
1.	"$\overrightarrow{SL}_{rnd}$" calls other search agents by using Equation 11.
2.	**For** each specific search agent (**j**)
3.	Update "$\vec{B}$"
4.	Update "$\vec{C}$"
5.	Update "$\overrightarrow{SL}$"
6.	Compute "$\overrightarrow{SP}_{leader}$" by using Equation 11.
7.	Compute the distance between each search agent (j) and "$\overrightarrow{SL}_{rnd}$" by using Equation 13.
8.	**If** "$\overrightarrow{SP}_{leader}$" < 0.25
9.	Update the location of the current search agent (**j**) by using Equation 9.
10.	**Else If** |**C**| < 1
11.	Update the location of the current search agent by using Equation 13.
12.	**Else**
13.	Select a new random search agent "$\overrightarrow{SL}_{rnd}$". // The selection is done randomly.
14.	Update the location of the current search agent (**j**) by using Equation 12.
15.	**End if**
16.	**End If**
17.	**End For** // The final stage

Fig. 12. Fitness function for updating the positions of sea lions to join a cluster

Description	This algorithm is used to cluster and calculate the local maximum flow for each specific cluster. The procedure proceeds recursively until one or more of the termination conditions (i.e. termination criterion) are reached that were previously stipulated by the user.
	Method
1.	**For** each cluster of the "N" clusters
2.	Choose "$\overrightarrow{SL}_{rnd}$" randomly // Initialization
3.	**Repeat** // Repeat while the termination criterion is not attained.
4.	Call the fitness function. // See Fig. 11 for the detail of this function.
5.	Call the MFP function. // See Fig. 14 for the detail of this function.
6.	**Until** (**t** ≥ Max-iteration) // The stopping condition (i.e. termination criteria) is met.
7.	**Return** the Local Maximum Flow (i.e. $maxFlow^{local}$) for each cluster. // This is the output of the algorithm.
8.	**End For**// The final stage

Fig. 13. The pseudocode for the cluster function

Description	This algorithm is used for obtaining the maximum flow from the beginning source vertex to the objective sink vertex.
	Method
1.	Choose all the routes from the source (i.e. the sea lion) to the destination (i.e. the prey).
2.	**For** each arc (a, b) in the graph of interest: // repeat as long as there is an edge "$e_{ij} \in E$".
3.	Arc's flow= 0
4.	**While** there is a route from sea lion to the prey in the graph.
5.	RemainingCapacity (P) = Min (RemainingCapacity (a, b) for the arc in the rout.
6.	Flow= Flow + RemainingCapacity (P)
7.	**For** each specific arc in the rout.
8.	**If** (a, b) is a forward arc
9.	Flow (a, b) = Flow (a, b) + RemainingCapacity (P)
10.	**Else**
11.	Flow (a, b)= Flow (a, b) - RemainingCapacity (P)
12.	**End If**
13.	**End For**
14.	**End While**
15.	**End For**
16.	**Return** MF // This is the output of this algorithm.

Fig. 14. Maximum Flow Problem (MFP) function

7. Results, Data Analysis, and Comparison

Motivated by the mission of evaluating the overall performance of the proposed solution, the SLnO-MFP algorithm was conducted and tested using a dataset from various scales of flow network sizes. These sizes were basically determined based on the previous research studies found in the literature which have dealt with similar situations, such as [24], [49], [1], [50], and [40].

Since it is always effective to evaluate the efficacy of a new approach being tested by comparing it with the already existing ones, the FF technique has been selected here as a baseline to compare the efficacy of the proposed algorithm, SLnO-MFP. It is also important to point out that the experiments of this comparison were conducted on the same datasets using the same situation related to the underlying resources, operating system (OS), settings, and specifications. Therefore, all the experiments are programmed in MATLAB® and carried out using a personal computer with Intel Core i-7 processor @2.4GHz, 16 GB RAM, and a 64-bit Microsoft Windows 10 operating system.

The evaluation metrics that were used in this paper are the average execution time, speedup, Relative Estimation Error rate (REE), Mean Relative Estimation Error (MREE), Accuracy (Acc), Mean Square Error (MSE), and deviation of the MSE (DMSE). These metrics were selected carefully because they are commonly used in the optimization problems, especially the MFP.

7.1 Run time results

According to the experimental analysis, the overall performance of the proposed algorithm, SLnO-MFP, and the Ford-Fulkerson (FF) algorithm are compared with each other using the same dataset size. Accordingly, Table V represents the average CPU's computational run time for both algorithms for each specific dataset's size. Each row of this table represents a given network size.

From a purely mathematical modeling angle, the relative speed-up (S^{speed}) of execution for each dataset's size is calculated and reported according to Equation 16:

$$RS^{speedup} = \frac{\text{Average run-time of FF}}{\text{Average run-time of SLnO-MFP}} \quad (16)$$

As shown in Table V, the average speedup (Speedup^{average}) of execution is calculated by finding the row-wise summation of the numeric values of the speedup (i.e. last column) multiplied by the network size (i.e. second column) of every experiment, and then dividing by the total number of the network sizes (i.e. summation of the second column). This is expressed in Equation 17:

$$\text{RSpeedup}^{average} = \frac{\sum_{j=1}^{M}\left(S_j^{\text{speedup}} * net_j^{size}\right)}{\sum_{j=1}^{M} net_j^{size}} \quad (17)$$

Where "net_j^{size} ", "M", and "$S_j^{speedup}$ " represent the $\mathbf{j^{th}}$ size of the network, the count of the elements in the data set, and the speedup of the $\mathbf{j^{th}}$ dataset's size which was computed as indicated by Equation 16, respectively. It is to be noted that the reported average speedup of the processing is comparatively recorded to be (7.4902) faster than FF. Hence, this result proves that this proposed algorithm is highly comparable in terms of the execution time.

Table V. An execution time comparison between SLnO-MFP and FF

j	net_j^{size}	Average run-time of FF (per seconds)	Average run-time of SLnO-MFP (per seconds)	Relative Speedup SLnO-MFP is faster than FF by:
1	**100**	**0.159**	**0.051**	**3.1176**
2	**200**	**0.312**	**0.072**	**4.3333**
3	**300**	**0.485**	**0.082**	**5.9146**
4	**400**	**0.698**	**0.091**	**7.6703**
5	**500**	**0.749**	**0.116**	**6.4569**
6	**600**	**0.913**	**0.125**	**7.3040**
7	**700**	**1.216**	**0.142**	**8.5634**
8	**800**	**1.549**	**0.212**	**7.3066**
9	**900**	**2.260**	**0.255**	**8.8627**
10	**1000**	**2.402**	**0.310**	**7.7484**
			The average speedup	**7.4902**

In a nutshell, it is clearly observed from this comparison that the proposed model SLnO-MFP performs best and gives better performance results than FF in terms of speed and time efficiency; it is faster than FF by an average of (7.4902) times. Furthermore, there is a dramatic increase in the difference in speed between the two algorithms when large-sized network instances are used. It would be important to know that the perceived complexity is remarkably behind this speedup and has a strong influence on the implementation of any metaheuristic algorithms. As reflected in the plot of Fig. 15, the execution complexity is a quadratic polynomial. More precisely, it begins to mount when the number of nodes increases.

To make a further comparison and evaluation, the impact of the network sizes on the speedup of both algorithms where both the first and last columns of Table V are graphically depicted in Fig. 16 for both of the two algorithms.

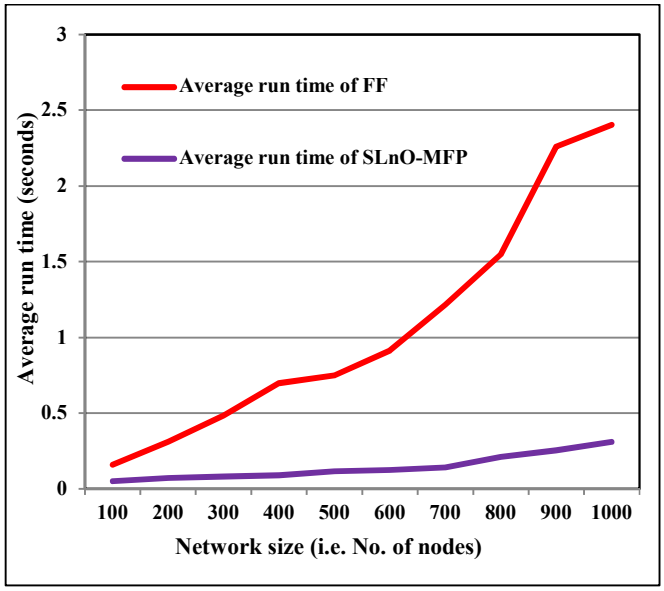

Fig. 15. The average CPU's computational run time for FF versus SLnO-MFP

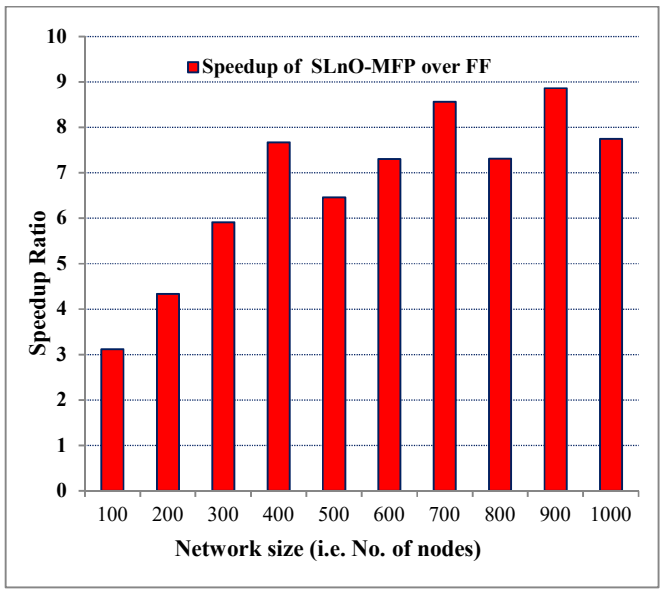

Fig. 16. The relative speedup of SLnO-MFP in comparison with FF algorithm

7.2 Relative Estimation Error Rate

Table VI illustrates the estimated-theoretical ($T^{estimated}$) and the actual-experimental (T^{actual}) run time of SLnO-MFP algorithm with a Relative Estimation Error rate (REE) which is calculated by using Equation 18:

$$\text{REE} = \frac{\text{abs}\left(\text{Estimated run time} - \text{Analysis run time}\right)}{\text{Analysis run time}} \quad (18)$$

Since the "analysis run time" represents "the estimated theoretical run time", Equation 18 can be redrafted as expressed in Equation 19:

$$REE = \frac{abs\ (Experimental\quad run\ time\ -\ Theoretical\quad run\ time\ )}{Estimated\quad theoretical\quad run\ time} = \frac{abs\ (T^{actual} - T^{estimated})}{T^{estimated}} \tag{19}$$

According to the statistical analysis of the sixth column of Table VI, it is seen that the proposed algorithm has low error rates compared to the FF algorithm.

Like the notion of Equation 17, the Mean Relative Estimation Error (MREE) is calculated and viewed in this table by summing up products between each element of the last column and the second column; then dividing this summation over the summation of the second column. This is expressed by using Equation 20:

$$\text{MREE} = \frac{\sum_{j=1}^{M}(REE_j * net_j^{size})}{\sum_{j=1}^{M} net_j^{size}} \tag{20}$$

Where "$\boldsymbol{REE_j}$", "$\boldsymbol{net_j^{size}}$ ", and "$\boldsymbol{M}$" represent the relative error of the $\mathbf{j^{th}}$ dataset's size which was computed as indicated by Equation 18, the $\mathbf{j^{th}}$ dataset's size of the network, and the count of the elements in the data set, respectively.

Table VI. Theoretical versus experimental Run-Time error of SLnO-MFP algorithm

j	net_j^{size}	Estimated theoretical run time (T)	Estimated experimental run time (E)	Error=abs(T-E)	REE_j=abs(T-E)/T	$REE_j * net_j^{size}$	$(REE_j)^2 * net_j^{size}$
1	**100**	**0.0929**	**0.0621**	**0.0308**	**0.3315**	**33.1500**	**10.9892**
2	**200**	**0.4785**	**0.0759**	**0.4026**	**0.8414**	**168.2800**	**141.5908**
3	**300**	**0.2762**	**0.0846**	**0.1916**	**0.6937**	**208.1100**	**144.3659**
4	**400**	**0.5998**	**0.9001**	**0.3003**	**0.5007**	**200.2800**	**100.2802**
5	**500**	**0.8672**	**0.9975**	**0.1303**	**0.1503**	**75.1500**	**11.2950**
6	**600**	**1.5341**	**0.1167**	**1.4174**	**0.9239**	**554.3400**	**512.1547**
7	**700**	**3.0921**	**0.1627**	**2.9294**	**0.9474**	**663.1800**	**628.2967**
8	**800**	**4.2753**	**0.1998**	**4.0755**	**0.9533**	**762.6400**	**727.0247**
9	**900**	**6.3782**	**0.2274**	**6.1508**	**0.9643**	**867.8700**	**836.8870**
10	**1000**	**8.3462**	**0.2681**	**8.0781**	**0.9679**	**967.9000**	**936.8304**
						MREE=0.8183	**MSE = 0.7363** **DMSE = 0.8581**

The result of this equation, namely Equation 20, was calculated using the seventh column of Table VI and then viewed in the last cell of the same column and table. Here again, it is noteworthy that the proposed algorithm is able to reduce the error rate to be (0.8183). Hence, this result proves that this proposed algorithm is highly comparable in terms of the error rate.

In an analogy with Equation 19, the average of the squared errors for all network sizes, which is referred to as the value of the Mean Square Error (MSE), was also selected to be an authentic validation measurement as shown in Equation 21:

$$\text{MSE} = \frac{\sum_{j=1}^{M}(REE_j)^2 * net_j^{size}}{\sum_{j=1}^{M} net_j^{size}} \tag{21}$$

The result of this equation was calculated using the last column of Table VI and then viewed in the last cell of the same column and table. By analyzing this calculated value, it is easily seen that the proposed model has achieved a very worthy result where the recorded MSE value is (0.7363). Remarkably, this slight difference occurs due to randomness inside the equations that are used in building the algorithm, like the ninth and the fourteenth equations. Yet again, this result also ensures that SLnO-MFP is highly comparable in terms of the MSE.

Furthermore, the last cell of Table VI is related to the deviation of the MSE, termed as (DMSE), which is calculated by taking the square root of the MSE. It should be noted that MSE and DMSE are calculated in the same way as the variance and standard deviation are usually computed, respectively. So, they have obviously the same unit type of measurement as in the case of the estimated quantities of variance and standard deviation.

Additionally, Fig. 17 exhibits a visualization of the enhancement that is accomplished in this work compared to the FF technique in terms of execution time. In view of this, it is relatively clear that the proposed technique has accomplished better performance especially for resolving the networks of large-sized instances.

7.3 Results Accuracy and Discussion

In order to examine the obtained results and to make a comparative analysis between the maximum flow value of the proposed algorithm SLnO-MFP and the FF algorithm, two major factors related to the overall performance evaluation are taken into consideration: the average execution time and the accuracy of the results. While the first evaluation of performance has been mentioned in the first subsection, the second evaluation is described here in this subsection.

As clearly viewed in the comparison of Table VII, the maximum flow values were calculated for both techniques using ten experiments, each has a different network size. After those experiments were conducted, the accuracy comparison for both techniques was calculated by using Equation 22 as a measure of accuracy (Acc):

$$Accuracy = \left(1 - \frac{abs(FF_{M\,FPvalue} - \mathbf{SLnO - M\,FP}_{M\,FPvalue})}{FF_{M\,FPvalue}}\right) * 100\% \quad (22)$$

Where "$FF_{M\,FPvalue}$" represents the maximum value using the FF technique and "$\mathbf{SLnO - M\,FP}_{M\,FPvalue}$" indicates the maximum value using the suggested algorithm (SLnO-MFP).

In order to compute the overall validation accuracy of SLnO-MFP algorithm, the average accuracy of all the network sizes in the dataset was calculated by utilizing Equation 23:

$$\text{The Overall accuracy} = \left(\frac{\sum_{j=1}^{M}(Acc_j * net_j^{size})}{\sum_{j=1}^{M} net_j^{size}}\right) \quad (23)$$

Where "net_j^{size}" represents the $\mathbf{j^{th}}$ the dataset's size, and the "Acc_j" is computed as indicated by Equation 22. The result of this equation was calculated and then viewed in the last row of Table VII. As represented in this comparative analysis, the proposed algorithm is able to attain a high accuracy percentage of (94.3205%). It is observed that the main intuition behind the overall accuracy relates to the difference between the maximum value for the proposed algorithm and the maximum value for FF technique which comes from the way of catching the network graph in the search space where the SLnO-MFP technique divides the network graph into a number of subgraphs.

These outstanding findings prove that the algorithm has superior performance and it is a viable alternative algorithm that can be efficiently used to solve many optimization problems of large-scale sizes, as in the case of this underlying problem (MFP).

As a final point, the empirical evidence substantiates that this proposed algorithm is proportionally scaling with the problem size, in both memory and time. The interesting interpretation behind this issue is highly associated with network complicity. Anyway, this solution can be applied successfully to other styles of optimization problems, and the outcomes presented here have far-reaching consequences in many other domains.

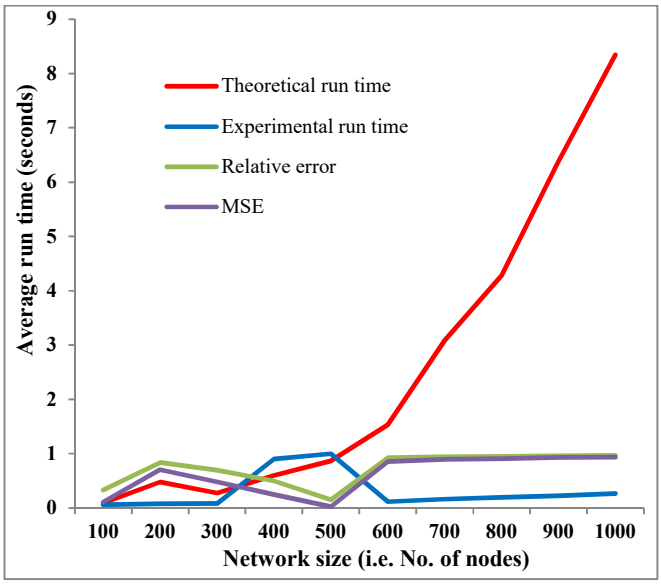

Fig. 17. FF versus SLnO-MFP in terms of execution time

Table VII. The accuracy results of SLnO-MFP compared with FF algorithm

j	net_j^{size}	FF-MFP (E)	SLnO-MFP (F)	abs(E- F)	abs(E – F) / E	Accuracy= 1- abs(E – F) / E	Accuracy percentage= (1- (abs(E – F) / E) * 100%
1	100	718	758	40	0.0557	0.9443	94.4290%
2	200	889	899	10	0.0112	0.9888	98.8751%
3	300	1873	1985	112	0.0598	0.9402	94.0203%
4	400	2465	2625	160	0.0649	0.9351	93.5091%
5	500	3167	3347	180	0.0568	0.9432	94.3164%
6	600	3708	4106	398	0.1073	0.8927	89.2665%
7	700	4568	4872	304	0.0665	0.9335	93.3450%
8	800	5097	5179	82	0.0161	0.9839	98.3912%
9	900	5826	6506	680	0.1167	0.8833	88.3282%
10	1000	6325	6346	21	0.0033	0.9967	99.6680%
The Overall accuracy						0.9432	94.3205%

8. Conclusions

Evidently, the intuition behind optimization is born out of the necessity for finding the best available solution among a set of candidate ones. When scientists make an insightful look behind the scenes of many creatures, they every day find numerous thought-provoking blips that can be used in overcoming a great portion of real-life optimization problems. Nowadays, swarm-intelligence-based (SIB) algorithms are one of the most Artificial Intelligence (AI) prevalent pillars which due to its remarkable advantages become an essential part of modern global optimization algorithms as well as the most widely implemented.

At its broadest, it is important to realize that optimization algorithms have their radical significance in the context of solving the Maximum Flow Problem (MFP). Besides that this added-value research paper covers and outlines the theoretical vision and the practical aspects of the metaheuristics, it introduces a population-based and nature-inspired metaheuristic algorithm to solve the Maximum Flow Problem (MFP) based on drawing inspiration from one of the core activities of the sea lions (SLn) which is the joint hunting behavior. As well, this paper can be used as a platform for selecting whichever approach is the best one out of the metaheuristics community.

After the research work reported in this paper, namely SLnO-MFP, has been applied, tested, and evaluated on various-scales datasets, the overall outcomes of have both theoretical and applied implications and the empirically-based results demonstrate that this algorithm is a highly-efficient in the long run and has a superior performance and competitive findings when compared with the other available optimization algorithms. Moreover, it is relatively clear that these viable outcomes in a way or another reinforce the algorithm's power to impose itself on solving the MFP.

According to the empirical analysis of the experiments, the overall performance of the proposed algorithm was compared with the Ford-Fulkerson (FF) algorithm. The worthy findings which have arrived with have shown that the proposed algorithm performs better compared to other similar research methods; it has attained a high accuracy percentage of (94.3205%) with an acceptable Mean Relative Estimation Error (MRE) rate of (0.8183), a Mean Square Error (MSE) of (0.7363), and an average speedup of (7.4902). Armed with these facts, these impressive experimental results give sufficient sound evidence and reinforce the conclusion that this proposed metaheuristic algorithm has far-reaching consequences in solving various real-world applications including the considered problem.

9. Future Work and Outlook

As the chronological progress poses new challenges and on the basis of the structure of the problem in hand, here are six vital pivots which need to be significantly addressed in the next few years as inspiring directions for further research and experimentation:

- **Parallel implementation:** From a purely practical standpoint, it will be more effective if the search time is reduced by applying the algorithm presented in this research work in a distributed parallel execution environment with an efficient dynamic clustering algorithm. Rather than implementing the whole MFP graph on a single processor, the graph is segmented into a number of independent partitions; each one is computed on a single processor. Then, the optimal maximum flow of these partitions is selected. [12][43][20]
- **Adaptive intelligent metaheuristics:** It is a timely stage where the other interested researchers can work to modify the proposed algorithm, SLnO-MFP, such that the parameters are more self-tuned during the running time according to the objective function values.
- **Optimization and performance metrics:** Rather than using a number of the classical ad-hoc statistical measurements that are utilized as assessment and comparison criteria for making performance measurements, improvement percentages, and function evaluations such as standard deviation, variance, correlation, skewness, and the simple mean, it is strongly recommended without reservation to establish a well-defined quantitative measuring framework that will be used as an in-depth assessment tool and an authentic criterion for efficiency validating of most metaheuristics [6][20].

  On the other hand, since most researchers, use the computational run-time of the CPU as their only primary resource for comparing the performance values of their algorithms, their perceptions should be extended to tackle other vital resources measurements that cannot be neglected or relegated, such that memory, and network bandwidth and other parameters that are used to measure the resources' efficiencies.

  Since the success of any optimization algorithm is topped by the active balancing between exploration and exploitation, there is also a need to establish a smart agreed criterion that guarantees this balancing ratio over the SLnO-MFP.
- **Convergence' acceleration and analysis:** Accelerating the objective function convergence has a great effect in raising the outcome efficiency and shrinking the needed number of generations necessary in arriving at the ultimate solution. Likewise, this might make the chance of arriving at the most-fit solution tend to be greater. Metaheuristics' convergence analysis of the objective functions has not still been fully addressed to reach a maturity level. Thus, another potential-open research area could be raised for further literature. [20]

- **Cloud Computing (CC):** Due to the voluminous amounts of today's data and on behalf of the Internet advances, CC applications are nowadays becoming more prevalent than it was a few years ago and, in turn, the most endurable place for hosting and activating optimization community. Since each cloud-based application should be premised on the faith that a large-pool of scalable, parallel, and distributed computing resources is granted to thousands and even millions of customers by the cloud vendor, it's time to go an important step forward to devote the efforts in integrating the metaphors of the metaheuristics to fully cope within the CC platform. This is particularly relevant to the case of hosting this presented algorithm. [12][43][51]
- **Hybridization:** Since it turns out that things work differently with hybridization of two or more exact, heuristic, or metaheuristic techniques, many promising outlets and opportunities for further research could be opened by using adaptive hybridization. From another direction, the degree of this adaptively should be used as a crucial tool that goes with the problem complexity. [20]

Acknowledgments

The author is grateful to WISE University, Amman, Jordan for the financial support granted to cover the publication fee of this research article. Secondly, the author would like to express his cordial thanks to Dr. Adel Hamdan, Eng. Nabeel Abuhamdeh, and Dr. Raja Masadeh in/for the great support and assistance rendered to carry out this research work. Finally, special gratitude to the editor and the honorable anonymous reviewers of IJCSNS for their perceptive comments, valuable suggestions, and magnificent efforts that helped the author to improve this paper.

References

[1] R. Masadeh, A. Alzaqebah, and A. Sharieh, "Whale Optimization Algorithm for Solving the Maximum Flow Problem," *Journal of Theoretical and Applied Information Technology (JATIT)*, vol. 96, no. 8, pp. 2208–2220, 2018.

[2] R. Masadeh, B. A. Mahafzah, and A. Sharieh, "Sea Lion Optimization Algorithm," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 5, pp. 388–395, 2019, doi: 10.14569/ijacsa.2019.0100548.

[3] P. Sindhuja, P. Ramamoorthy, and M. S. Kumar, "A Brief Survey on Nature Inspired Algorithms: Clever Algorithms for Optimization," *Asian Journal of Computer Science and Technology (AJCST)*, vol. 7, no. 1, pp. 27–32, 2018.

[4] P. F. Felzenszwalb and R. Zabih, "Dynamic Programming and Graph Algorithms in Computer Vision," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 4, pp. 1–51, 2010.

[5] P. Ryser-welch and J. F. Miller, "A Review of Hyper-Heuristic Frameworks," in *Electronic Village Online - Evo20 Workshop, American International School of Bucharest (AISB)*, 2014, pp. 1–7.

[6] M. Q. Al-shammari and R. C. Muniyandi, "Optimised Tail-based Routing for VANETs using Multi-Objective Particle Swarm Optimisation with Angle Searching," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 6, pp. 224–232, 2020.

[7] L. I. Ausiello G, Franciosa PG and R. A., "Max Flow Vitality in General and st-planar Graphs," *Networks*, vol. 74, no. 1, pp. 70–78, 2019, doi: 10.1002/net.21878.

[8] "Ford-Fulkerson Algorithm for Maximum Flow Problem," *Geeks for Geeks, a Computer Science Portal for Geeks*, 2020. https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/ (accessed Jul. 18, 2020).

[9] M. Al-Ta'ee, N. K. T. El-Omari, and W. Al Kasasbeh, *Information Systems Analysis and Design*, First edit. Amman, Jordan: Dar Al-Massira for Printing-Publishing, ISBN: 978-9957-069-483, pp.1-527, 2013.

[10] S. Consoli, "The Development and Application of Metaheuristics for Problems in Graph Theory: A Computational Study," School of Information Systems, Computing and Mathematics, Brunel University, West London, pp. 1-222, 2008.

[11] M. Y. Alkhanafseh, M. Qatawneh, and H. A. Ofeishat, "A Parallel Chemical Reaction Optimization Algorithm for MaxFlow Problem," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 15, no. 6, pp. 19–32, 2017.

[12] N. K. T. El-Omari, "Cloud IoT as a Crucial Enabler: a Survey and Taxonomy," *Modern Applied Science*, vol. 13, no. 8, pp. 86–149, 2019, doi: 10.5539/mas.v13n8p86.

[13] A. Lam and O. Victor Li., "Chemical Reaction Optimization: a tutorial," *Memetic Computing*, vol. 4, no. 1, pp. 3–17, 2012, doi: 10.1007/s12293-012-0075-1.

[14] A. Prakasam and N. Savarimuthu, "Metaheuristic Algorithms and Polynomial Turing Reductions: A Case Study Based on Ant Colony Optimization," in *International Conference on Information and Communication Technologies (ICICT), Karachi, Pakistan*, 2015, vol. 46, pp. 388–395, doi: 10.1016/j.procs.2015.02.035.

[15] C. B. Oscar, P. P. Parra, and B. Hernández Ocana, "On combining numerical optimization techniques with a belief merging approach," in *Eleventh Latin American Workshop on New Methods of Reasoning (LANMR), vol. 2264, paper 5*, 2018, pp. 51–62, doi: 10.1016/j.inffus.2016.02.006. 5.

[16] S. Mirjalili, S. M. Mirjalili, A. Lewis, C. Technology, and S. Beheshti, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[17] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, First edit. United States: Lulu Publishing, ISBN: 10-1446785068, ISSN: 978-1446785065, doi: 10.5281/zenodo.3566253, pp. 1-438, 2012.

[18] N. K. T. El-Omari, "A Hybrid Approach for Segmentation and Compression of Compound Images," *The Arab Academy for Banking and Financial Sciences*, pp. 1–201, 2008.

[19] G. Du, X. Liang, and C. Sun, "Scheduling Optimization of Home Health Care Service Considering Patients' Priorities and TimeWindows," *Sustainability*, vol. 9, no. 253, pp. 1–22, 2017, doi: 10.3390/su9020253.

[20] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic Research: a Comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2191–2233, 2019, doi: 10.1007/s10462-017-9605-z.

[21] A. Lam and O. Victor Li., "Chemical-Reaction-Inspired Metaheuristic for Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 381–399, 2010, doi: 10.1109/TEVC.2009.2033580.

[22] W. J. Hopp and M. L. Spearman, *Factory Physics*, Third Edit. United States: Waveland Press, ISBN:13:978-1577667391, ISSN: 10-577667395, pp. 1-746, 2008.

[23] Pierce Rod., "Polynomials," *Math is Fun*. https://www.mathsisfun.com/algebra/polynomials.html (accessed Jul. 17, 2020).

[24] R. Masadeh, A. Sharieh, and A. Sliet, "Grey wolf optimization applied to the maximum flow problem," *International Journal of Advanced and Applied Sciences*, vol. 4, no. 7, pp. 95–100, 2017, doi: 10.21833/ijaas.2017.07.014.

[25] Wikipedia Contributors, "Bibliographic details for 'Heuristic (computer science),'" *Wikipedia, The Free Encyclopedia*, 2019. https://en.wikipedia.org/wiki/Heuristic_(computer_science) (accessed Jul. 17, 2020).

[26] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[27] Z. H. Ahmed, "A Comparative Study of Eight Crossover Operators for the Maximum Scatter Travelling Salesman Problem," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, pp. 317–329, 2020, doi: 10.14569/IJACSA.2020.0110642.

[28] R. "Mohammad T. Masa'deh, "New Sea Animal Inspired Metaheuristic Approach for Task Scheduling in Cloud Computing," Department of Computer Science, The University of Jordan (UJ), pp. 1-242, 2019.

[29] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A Survey on New Generation Metaheuristic Algorithms," *Computers and Industrial Engineering, Elsevier*, vol. 137, no. 106040, pp. 1–69, 2019, doi: 10.1016/j.cie.2019.106040.

[30] Abu Doush Iyad *et al.*, "Harmony Search Algorithm for Patient Admission Scheduling Problem," *Journal of Intelligent Systems Harmony*, vol. 29, no. 1, pp. 1–25, 2018, doi: 10.1515/jisys-2018-0094.

[31] S. M. Saab, N. K. T. El-Omari, and H. H. Owaied, "Developing Optimization Algorithm Using Artificial Bee Colony System," *Ubiquitous Computing and Communication Journal*, vol. 4, no. 5, pp. 15–19, 2009.

[32] H. Pirim, E. Bayraktar, and B. Eksioglu, *Tabu Search: A Comparative Study*, 1st Editio. INTECH OPEN LIMITED, ISBN: 9783902613349, ISSN: 03038467, PMID: 27022619, pp. 1-27, 2008.

[33] Francisco Sáez, "Productivity Strategies: Exploration vs Exploitation," *Facile Things Blog*, 2020. https://facilethings.com/blog/en/exploration-vs-exploitation (accessed Sep. 11, 2020).

[34] R. Barham, A. Sharieh, and A. Sliet, "Chemical Reaction Optimization for Max Flow Problem," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 8, pp. 189–196, 2016, doi: 10.14569/ijacsa.2016.070826.

[35] N. K. T. El-Omari, A. H. Al-Omari, A. M. H. Al-ibrahim, and T. Alwada, "Text-Image Segmentation and Compression using Adaptive Statistical Block Based Approach," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 6, no. 4, pp. 1–9, 2017.

[36] N. K. T. El-Omari, A. H. Omari, O. F. Al-badarneh, and H. Abdel-jaber, "Scanned Document Image Segmentation Using Back-Propagation Artificial Neural Network Based Technique," *International Journal of Computers and Communications*, vol. 6, no. 4, pp. 183–190, 2012.

[37] S. Alghyaline, N. K. T. El-Omari, R. M. Al-Khatib, and H. Al-Kharbshh, "RT-VC: an Efficient Real-Time Vehicle Counting Approach," *Journal of Theoretical and Applied Information Technology (JATIT)*, vol. 97, no. 7, pp. 2062–2075, 2019.

[38] P. D. P. Reddy, V. C. V. Reddy, and T. G. Manohar, "Whale Optimization Algorithm for Optimal Sizing of Renewable Resources for Loss Reduction in Distribution Systems," *Renewables: Wind, Water, and Solar*, vol. 4, no. 1, pp. 1–13, 2017, doi: 10.1186/s40807-017-0040-1.

[39] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *The Canadian Journal of Mathematics (CJM)*, vol. 8, no. 3, pp. 399–404, 1956.

[40] R. Masadeh, A. Alzaqebah, B. Smadi, and E. Masadeh, "Parallel Whale Optimization Algorithm for Maximum Flow Problem," *Modern Applied Science*, vol. 14, no. 3, pp. 30–44, 2020, doi: 10.5539/mas.v14n3p30.

[41] O. M. Surakhi and H. A. Ofeishat, "A Parallel Genetic Algorithm for Maximum Flow Problem," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 6, pp. 159–164, 2017.

[42] R. L. Brandt, "Jordan Unveils PS3-based Supercomputer," *High Performance Computing (HPC), HPC Wire*, 2020. https://www.hpcwire.com/2013/03/07/jordan_s_25_teraflop_playstation_3/ (accessed Jun. 12, 2020).

[43] N. K. T. El-Omari, "An Efficient Two-level Dictionary-based Technique for Segmentation and Compression Compound Images," *Modern Applied Science*, vol. 14, no. 4, pp. 52–89, 2020, doi: 10.5539/mas.v14n4p52.

[44] CP-Algorithms, "Maximum flow - Ford-Fulkerson and Edmonds-Karp," 2020. https://cp-algorithms.com/graph/edmonds_karp.html (accessed Aug. 01, 2020).

[45] Wikipedia Contributors, "Bibliographic details for 'Swarm Behaviour,'" *Wikipedia, The Free Encyclopedia*, 2020. https://en.wikipedia.org/wiki/Swarm_behaviour (accessed Jul. 05, 2020).

[46] B. N. Vachaku, "A Reflective Swarm Intelligence Algorithm," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 14, no. 4, pp. 44–48, 2013.

[47] Wikipedia Contributors, "Bibliographic details for 'Sea lion,'" *Wikipedia, The Free Encyclopedia*, 2020. https://en.wikipedia.org/w/index.php?title=Sea_lion&oldid=966030165 (accessed Jul. 25, 2020).

[48] Wikipedia Contributors, "Bibliographic details for 'Speed of sound,'" *Wikipedia, The Free Encyclopedia*, 2020. https://en.wikipedia.org/w/index.php?title=Speed_of_sound&oldid=968529357 (accessed Jul. 24, 2020).

[49] A. Alzaqebah, R. Masadeh, and A. Hudaib, "Whale Optimization Algorithm for Requirements Prioritization," in *International Conference on Information and Communication Systems (ICICS), Irbid, Jordan*, 2018, pp. 84–89, doi: 10.1109/IACS.2018.8355446.

[50] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Algorithm for Requirements Prioritization," *Modern Applied Science*, vol. 12, no. 2, p. 54, 2018, doi: 10.5539/mas.v12n2p54.

[51] N. K. T. El-Omari and M. H. Alzaghal, "The Role of Open Big Data within the Public Sector, Case Study: Jordan," in *The 8th International Conference on Information Technology (ICIT 2017), IEEE, Amman, Jordan, doi: 10.1109/ICITECH.2017.8079997*, 2017, pp. 182–186.

Author's Profile

Nidhal K. T. El-Omari received his B.Sc. in Computer Science and his M. Eng. degree in Computer Engineering in 1986 and 2005, respectively, both from Yarmouk University, Irbid-Jordan. In 1989, he received his Higher Diploma of Branch Automation Officer from the Department of Defense (DoD), Fort Gordon / Georgia-USA. In 2008, he received a doctorate in Computer Information Systems and Image Processing from The Arab Academy for Banking and Financial Science (AABFS), Amman-Jordan.

He joined the Information Technology Directorate of the Jordanian Ministry of Defense in 1986 and retired in 2009. During those 24 years, he chaired a number of IT-related departments including the Systems Follow-up Department, Technical Support Department, and Automation Department. He has been at the Faculty of IT since 2009, WISE University, during which he worked as the director of the Computer Center, the Chair of the Department of Computer Science and Basic Science, and the head of the Department of Software Engineering. Since 2015, he is an Associate Professor. His research interests include, but are not limited to, image compression & segmentation, evolutionary computation, heuristic optimization, and methodologies for building both secure and strategical-efficient software. Dr. El-Omari has authored/co-authored two computer books and published more than thirty research articles and conference papers in top-quality journals and conference proceedings. By last, he can be reached via e-mails at nidhal.omari@wise.edu.jo or omari_nidhal@yahoo.com