

A Compendium over Nature-Inspired Metaheuristic Algorithms for Prioritization of Requirements

Neha Mishra

Department of Computer Science, SRIMT, Lucknow, India

Email- nehabtcs@gmail.com

Jitesh P. Tripathi*

P.G. Department of Mathematics, R. N. College, Hajipur

(B. R. A. Bihar University, Muzaffarpur), India

*Corresponding author- jiteshmaths@gmail.com

(Received August 8, 2022, Accepted August 30, 2022)

Abstract: In the process of requirement engineering prioritize the requirements is one of the most prominent tasks that are dominantly used in developing the project from the stockholders' viewpoint. Requirement engineering is considered as the base of software project development that relies on the systematic arrangement of requirements. Requirement prioritization is related to the managing of requirements from the stakeholder's perspective to cope up with certain constraints. When metaheuristic algorithms are applied in software engineering problems it is considered search-based software engineering. Metaheuristic algorithms have been used for requirements selection and prioritization to find the best result in the form of requirements that meet the need of user requests amid constraints such as cost, time, and limited resources. This paper encapsulates the classification and review of different nature-inspired metaheuristic algorithms that are used to prioritize the requirements till now and a comparative analysis is presented.

Keywords: Requirements, Prioritization, Metaheuristic Algorithm.

1. Introduction

Software Engineering (SE) comprises the most noteworthy field which is discern as the Requirement Engineering (RE). In the Software Development Life Cycle (SDLC) RE has taken as the most notable stage of the life cycle of product development. RE incorporates, elicitation, and validation of requirements and required documentation in its life cycle. The entire process of software project development process has several curtailments such as

budget and time, hence the release of the continuous version of a software project is necessary. Stakeholders participate in the selection process for the project. For a large project, it is difficult to make decisions as the project has many stakeholders to make decisions. This problem leads software engineers to obtain the feasible or correct requirements to build the right decision for the successful release of a software project.

1.1 Definitions of Software Requirements according to IEEE are defined as¹-

- (1) A circumstance or functionality wished with the aid of a person to clear up a problem or obtain an objective.
- (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents;
- (3) A documented representation of a condition or capability as in (1) or (2).

Requirements are not limited to the functionality of the system, as often supposed, but include other aspects. Different definitions have been presented by different authors. There are both functional and non-functional requirements in the system.

Davis² classifies requirements as:

- Functional requirements
- Nonfunctional requirements
- Performance / Reliability
- Interfaces
- Design constraints

In the software engineering process, different techniques were indulging in the selection of precise requirements. Numerous techniques are encapsulated for requirement prioritization. Few of them work great with projects having many requirements, while other techniques are incapacitated to give accepted outcomes. In the case of rigid execution plan, scanty resources, and customer's high-level prospects, then the most prominent characteristics should be published in early enough time³. This reason leads us to perceive the epochal of priority ordering of the requirements⁴.

1.2 Requirement Prioritization

Prioritization of requirements is the process of obtaining the correct or feasible set of requirements that compel stake holder's need and importance of release, for this purpose many algorithms are used. Sommerville defined the requirements prioritization as one of the most important tasks for

decision-makers⁵ While; Firesmith defines it as the weighty task in software engineering as it gives spanning implementation order of the requirements to mapping software versions and supplying adored functionality as early as possible or the process to define the priority of the requirements to stakeholders based on the importance of the requirements⁶. The extant requirement prioritization techniques reward the stakeholder's high chance to be involved in deciding which requirements reckoned on in a software based on their exigent. The stakeholders to transact with restrictions and limitations of the project's resources³. Moreover, negotiate the jarring between viewpoints that affect the software development. Unambiguous or precise requirements are the most prominent factor that affects the success of a project. Different metaheuristic or heuristic techniques like Genetic Algorithms (GA), Simulated Annealing (SA), Tabu Search (TS), ACO, PSO, Gray Wolf Optimization, Whale Optimization, Firefly Optimization and other are used in software engineering problem to solve the optimization, scheduling, prioritization (test case, requirements, etc.) and many other problems.

2. Reated Work

In the previous many years, various procedures have been created or raised to focus on the given prerequisites for a specific venture. A portion of these strategies are fitter to apply when there is few prerequisites and others are more exact for the mind boggling and tremendous number of necessities. The stakeholder plays an essential role in decision making to analyze the requirements. The most popular requirement prioritization (RP) techniques are classified into three categories⁷⁻⁹; Nominal, Ordinal, and Ratio Scale. Nominal Scale prioritization techniques produce several categories to which objects can be classified⁸. In other words, the requirements can be categorized into classes based on their exigence. MoSCoW (Must have, should have, could have, and will not have) and Numeral assignment techniques are only the techniques that are included in this category. Whereas Ordinal Scale prioritization techniques generate sorted and refined lists of requirements. These techniques can only say that one requirement is more significant than others but not to what extent. Priority group, bubble sort, and minimal spanning tree (MST) techniques are the most common and count in this classification. Ratio Scale prioritization techniques generate ordered lists of requirements too, but this category can supply the relative diversity. Hierarchical cumulative voting (HCV), Analytic Hierarchy Process (AHP), and Hierarchy AHP are the common techniques and count in this classification⁹.

The numerical assignment technique is recognized as the most common and traditional technique in prioritization process. In this technique many groups of requirements are created and on basis of priority each requirement is assigned to that group, but this doesn't depict which requirement has more or less primacy than other requirements within the same preponderance⁸. This technique is mentioned by many researchers such as Sommerville and Sawyer⁴, Karlsson *et al.*⁸, and Berander and Andrews¹⁰.

The 'Priority Groups' is another technique that is like the Numerical Assignment Technique, which distribute each of the given requirements in one of the three main categories; first is low class, second is medium class and the third is high class. However, the distinction between these mechanisms is that the numerical assignment technique classifies the requirements once, but the priority groups mechanism categorizes the requirements over and over again⁸⁻⁹.

Karlsson *et al.*⁸ assume the empirical study about these priority classes, which prioritize 13 requirements with these techniques; AHP, MST, binary search, bubble sort, and priority groups. This study draws that the worst technique is the priority groups' technique in terms of difficult to use and slow to implement. Thus, this technique is not suitable to be applied to a small number of requirements. While⁸ concludes that the AHP technique is time-consuming. Based on this conclusion, many techniques are developed to solve this issue by reducing the number of comparisons. Hierarchy AHP and MST are developed to improve the time-consuming problem. The bubble sort mechanism is compared in this study⁸ which is the simplest to use and supplied fault tolerance, as well as reliable. It is concluded that this technique is faster than AHP in terms of time consumption, but is slower than MST technique. Thus, the Bubble sort mechanism is suitable for a small number of requirements. Based on this study⁸, the MST is the fastest technique but it is not suitable for a small number of requirements. Many comparative studies of these techniques have been conducted to find the best prioritization technique.

Besides these techniques, many other AI Optimization Algorithms have also been used for Requirement Prioritization.

Paolo Tonella, Angelo Susi, Francis Palma¹¹ employed a Genetic algorithm for prioritizing the requirement. Masadeh *et al.*³ employed the grey wolf optimization algorithm for prioritizing the requirement; this algorithm mimics the hunting behavior of the grey wolf. Masadeh *et al.*¹² has also presented a research paper over the prioritization of requirements using the Whale optimization algorithm that is based on the bubble-net hunting technique of humpback whales.

Some other hybrid techniques are also developed for prioritizing the requirements.

3. Classification of Metaheuristic Algorithms

"Metaheuristic" was originated by Fred Glover in 1986, and it can be considered as a "master strategy that guides and modifies other heuristics to provide a sufficiently good solution especially with incomplete or imperfect information or limited computation capacity. Metaheuristic Algorithms are higher-level strategies to produce the best solution by exploring the search space. Metaheuristic Algorithms incorporate the diversification and intensification strategy: Diversification means generating diverse solutions to explore the search space on a global scale, while intensification means to focus the search in a local region to avoid the risk of knowing that a current good solution is found in a local region.

Properties that characterize Metaheuristic Algorithms:

1. Metaheuristic Algorithms are strategies that "guide" the search process.
2. The goal is to efficiently explore the search space to find near-optimal solutions.
3. Techniques that constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
4. Metaheuristic algorithms are approximate and usually non-deterministic.
5. Metaheuristic algorithms are not problem-specific
6. Metaheuristic algorithms may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper-level strategy.

Several Metaheuristic techniques may be classified based on their traits. In this paper, seven metaheuristic strategies are encapsulated with their evaluation.

Classification of Metaheuristic Optimization Algorithm is as follows.

3.1 Classification Based On Local and Global Search

Many metaheuristic algorithms were proposed to improve the local search heuristic to find better solutions. Such metaheuristic incorporates simulated annealing, tabu search, iterated local search, variable neighbourhood search, and GRASP. These metaheuristic algorithms can both be classified as local search-based or global search metaheuristic algorithms.

Other global search metaheuristic algorithms that are not local search-based are usually population-based metaheuristic algorithms. Such metaheuristic algorithms include ant colony optimization, evolutionary computation, particle swarm optimization, and genetic algorithms.

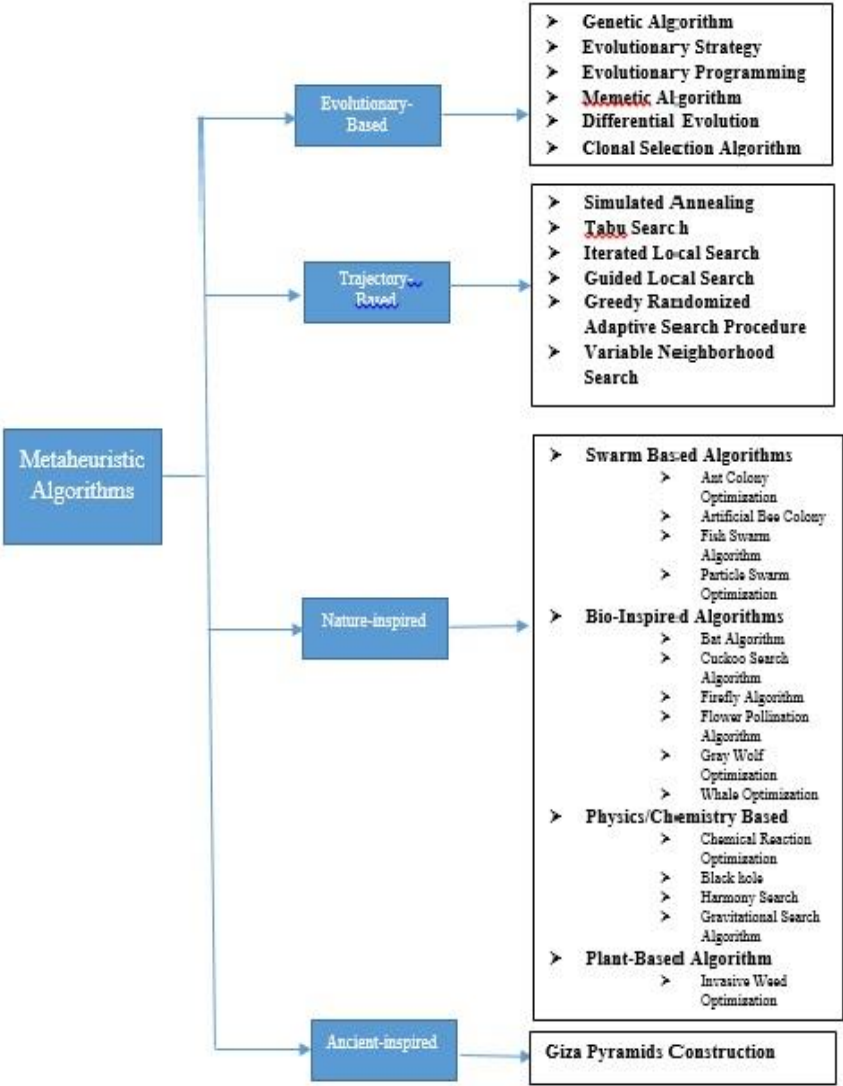


Figure 1. Classification of Metaheuristic Optimization Algorithm

3.2 Classification Based on Single Solution and Population

This type of classification is based on how many solutions at any given time by a metaheuristic algorithm. Population-based metaheuristic algorithms manipulate a set of solution and can find the best solution by using selection and combination of existing solution from the set whereas the single solution-based metaheuristic algorithms (trajectory-based algorithms) uses a one solution at a time. Single solution metaheuristic algorithms include simulated annealing, iterated local search, variable neighbourhood search, and guided

local search. Population-based metaheuristic algorithms include evolutionary computation, genetic algorithms, and particle swarm optimization. Another category of metaheuristic algorithms is Swarm intelligence which is a collective behaviour of decentralized, self-organized agents in a population or swarm. Ant colony optimization, particle swarm optimization, social cognitive optimization are examples of this category.

3.3 Classification Based on Nature-Inspired

In recent years, a very active area of research is the nature-inspired metaheuristic algorithms. These algorithms are derived from the behaviour of nature. Such metaheuristic algorithms include simulated annealing, evolutionary algorithms, ant colony optimization, and particle swarm optimization, Gray wolf optimization, Firefly optimization, Whale optimization, etc.

4. Nature-Inspired Meta-Heuristic Algorithms

4.1 Genetic Algorithm (GA)

The Genetic algorithm (GA) is developed by John Holland and his collaborators in the 1960s and 1970s is a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection. Holland was the first to use crossover and recombination, mutation, and selection¹³.

GA makes use of mutation, crossover, and selection operator for optimization methods. GA is a population-based method, *i.e.* Random amount of population of solutions termed as chromosomes. The fitness of every chromosome is determined by way of comparing it in opposition to an objective characteristic¹⁴. Using crossover and mutation method chromosomes are generated for the next generation (infant chromosomes) through previous new release's chromosomes (parent chromosomes).

In the crossover procedure, part of one of the parent chromosomes is blended with some other part of an exclusively determine chromosome generate an infant chromosome¹⁵.

In mutation technique price of determining chromosome are altered to generate child chromosomes after which new toddler chromosomes are evaluated for the objective function of the method. This method continues until the high-quality fit answer is received that needs to be close to the most effective solution.

Genetic algorithm has many advantages over traditional optimization algorithm and two noticeable advantages are the ability to deal with complex combinatorial problem and parallelism.

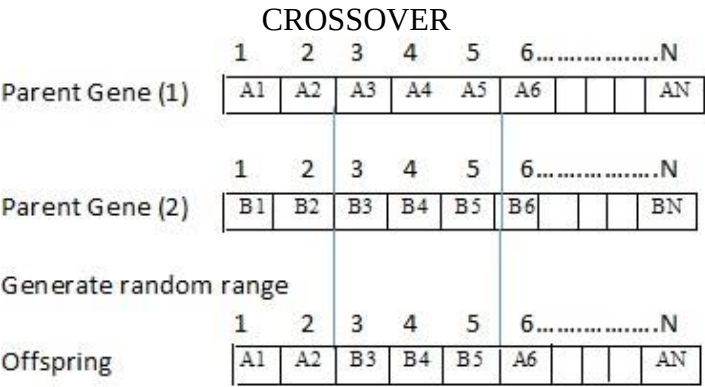


Figure 2. Crossover operation to generate offspring

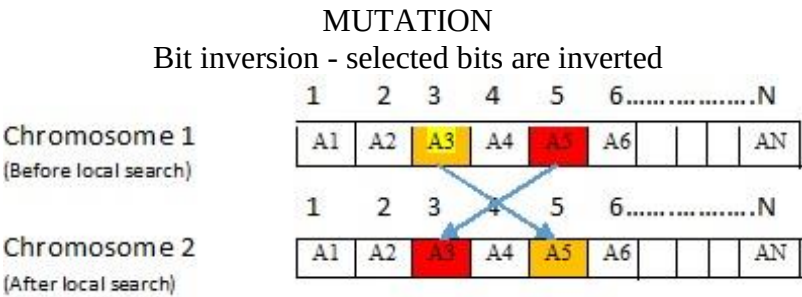


Figure 3. Applying local search using the pair-wise interchange

Pseudo-code for GA procedure¹⁴

```
Begin;  
Generate a random population of P solutions (chromosomes);  
For each individual i2P: calculate fitness (i);  
For i=1 to number of generations;  
Randomly select an operation (crossover or mutation);  
If crossover;  
Select two parents at random ia and ib;  
Generate on offspring ic = crossover (ia and ib);  
Else If mutation;  
Select one chromosome i at random;  
Generate an offspring ic = mutate (i);  
End if;  
Calculate the fitness of the offspring ic;
```

If i_c is better than the worst chromosome then replaces the worst chromosome by i_c ;
Next i ;
Check if termination=true;
End;

4.2 Grey Wolf Optimization Algorithm:

Grey Wolf Optimization (GWO) algorithm is one of the newly developed nature-inspired metaheuristic algorithms proposed by Mirjalili and Lewis in 2014¹⁶. The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Four types of grey wolves such as alpha, beta, delta, and omega are employed for simulating the leadership hierarchy. Besides, the three main steps of hunting, searching for prey, encircling prey, and attacking prey, are implemented¹⁶.

The main objective of this algorithm is to achieve an optimal solution for a given problem by utilizing the search agent's population.

In nature, these wolves live in packs; each pack size is 5-12 on average. The main distinction between GWO and other meta-heuristic optimization algorithms is the social dominant hierarchy whose main purpose is to develop the candidate solution during each iteration. Thus, the GWO imitates the hunting behavior of grey wolves in terms of finding and attacking prey¹⁶.

The social hierarchy of wolves and their responsibility is depicted in the following figure.

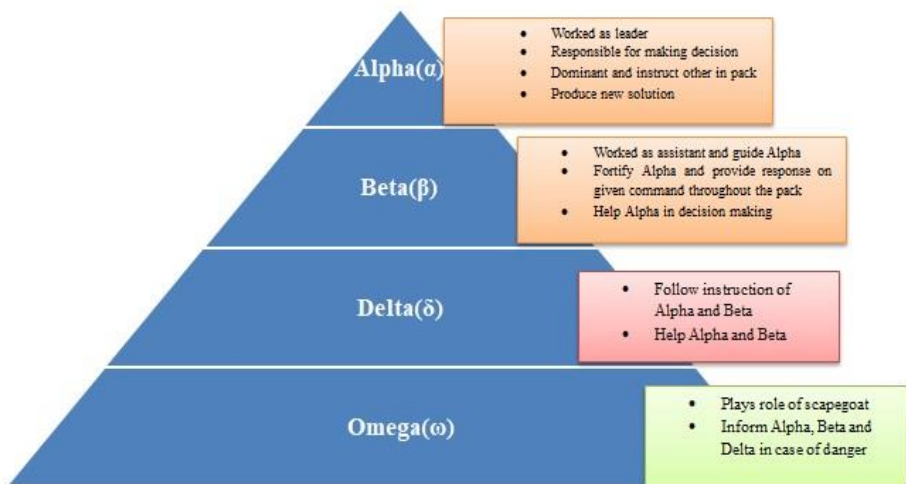


Figure 4. Hierarchy of Grey Wolf

Based on the Proposed algorithm by Mirjalili et al., 2014¹⁶, each level includes the following activity:

- The gray wolf in the first level of the hierarchy is called Alpha (α). They are leaders and responsible for making decisions and hence exhibit the best candidate solution.
- The Gray wolf in the second level of the hierarchy is called beta (β). They are responsible to assist the upper level i.e. Alpha to make decisions and act as a bridge between the higher level which contains alpha wolves and the lower level that includes the rest of the wolves in the pack. Hence, they exhibit the second-best candidate solutions.
- The third level contains Delta wolf (δ) that are responsible for conveying information to the higher level i.e. Alpha(α) and Beta(β) and exhibit the third best candidate solution.
- The last level wolves are Omega (ω) that are delivering the information to the higher level (α, β, δ). In this hierarchy, Omega plays the role of scapegoat and if they are not present then the situation of internal fighting may occur.

In addition to the social hierarchy of wolves, group hunting is another interesting social behaviour of grey wolves. Main phases of grey wolf hunting are as follows:

- Tracking, chasing and approaching the prey
- Pursuing, encircling, and harassing the prey until it stops moving
- Attack towards the prey

Thus, GWO acts as the foraging technique of the grey wolves mathematical which is used to solve the complex optimization problem. The optimal solution is considered as prey for the given issue. The three upper levels motions emulate the prey surrounding by grey wolves, which is the following mathematical equation is proposed in this regard¹⁶:

Encircling prey:

$$(1) \quad \vec{D} = \left| \vec{C} \cdot \vec{X}_p(t) - \vec{X}(t) \right|, \text{ with } \vec{C} = 2 \cdot \vec{r}_2$$

where t indicates the current iteration, X_p displays the prey position vector, X represents the grey wolf position, C is a coefficient vector and r_2 is denoted as a random vector in $[0, 1]$. Thus, the finding of vector D is employed to shift the specific element toward or far from the region that the best solution is placed that indicates the prey by applying the following formula¹⁶:

$$(2) \quad \vec{X}(t+1) = \left| \vec{X}_p(t) - \vec{A} \cdot \vec{D} \right|, \text{ with } \vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$$

where r_1 is chosen randomly in $[0, 1]$ and a is decreased from 2 to 0 over the range of iterations. In case $|A| < 1$, this means the exploitation phase is matched and emulates the behavior of assaulting the prey. While in case $|A| > 1$, this means that the exploration phase is matched and simulates the grey wolves spacing from the prey. The proposed values of A are in the range $[-2, 2]$. Thus, calculating the following mathematical equations to identify the top three levels α , β and δ ¹⁶:

Hunting:

$$(3) \quad \vec{D}_\alpha = \left| \vec{C}_1 \cdot \vec{X}_\alpha - \vec{X} \right| \text{ with } \vec{X}_1 = \vec{X}_\alpha - \vec{A}_\alpha \cdot (\vec{D}_\alpha)$$

$$(4) \quad \vec{D}_\beta = \left| \vec{C}_2 \cdot \vec{X}_\beta - \vec{X} \right| \text{ with } \vec{X}_2 = \vec{X}_\beta - \vec{A}_\beta \cdot (\vec{D}_\beta)$$

$$(5) \quad \vec{D}_\delta = \left| \vec{C}_3 \cdot \vec{X}_\delta - \vec{X} \right| \text{ with } \vec{X}_3 = \vec{X}_\delta - \vec{A}_\delta \cdot (\vec{D}_\delta)$$

with the view to mathematically simulate the foraging process of wolves, let α , β , and δ know the potential location of the prey. Moreover, the first three top solutions obtained from Alpha, Beta, and Delta are stored and compel other agents to update their positions accordingly. The following mathematical expression is given by Mirjalili *et al.*, 2014¹⁶:

$$(6) \quad \vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$$

Pseudo-code of the GWO algorithm¹⁶

Initialize the grey wolf population X_i ($i=1, 2, 3, \dots, n$)

Initialize a , A , and C

Calculate the fitness of each search agent

X_α = the best search agent

X_β = the second-best search agent

X_δ = the third best search agent

while ($t < \text{max number of iterations}$)

for each search agent

Update the position of the current search agent by equation (6)

end for

Update a , A , C

Calculate the fitness of all search agents

Update X_α , X_β , X_δ

$t=t+1$
end while
 return X_{α}

4.3 Whale Optimization Algorithm:

WOA is one of the latest randomly optimization techniques by Mirjalili and Lewis in 2016¹⁷. This technique aims to imitate the hunting behavior of humpback whales by following the location of prey and encircle it. The hunting journey of humpback whales contains two main stages; the exploration stage and exploitation stage. Thus, during searching in the search space for prey, WOA deems the current best candidate solution as target prey until the best solution is known because the optimal one is unknown a priori. Based on that, the other whales update their locations according to the current best solution¹².

This behavior is modeled mathematically by the following Equation (7) and Equation (8)¹⁷:

$$(7) \quad \vec{D} = \left| \vec{C} \cdot \vec{X}^*(t) - \vec{X}(t) \right|$$

$$(8) \quad \vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \vec{D}$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}^*$ is the position vector of the best solution. $\vec{X}^*$ should be updated in each iteration accordingly to the best solution. The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows by Eq. (9) and Eq. (10)¹⁷:

$$(9) \quad \vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a}$$

$$(10) \quad \vec{C} = 2 \cdot \vec{r}$$

where $\vec{a}$ is linearly decreased from 2 to 0 throughout iterations (in both exploration and exploitation phases) and $\vec{r}$ is a random vector in $[0,1]$ which has a significant role to make each search agent reaching any position during its search space to ensure the exploration¹⁷.

The main hunting behavior of humpback whales is the bubble net technique¹⁸ which represents the exploitation strategy in the WOA. The following approaches are modeled for the foraging mechanism:

- **Shrinking encircling mechanism:** In this process, the value of A is minimized by minimizing the value of a according to Eq. (3), and the search agent updates its value within search space with the new value that is among the original location and the current best location.

- **Spiral updating position:** This mechanism firstly computes the distance between the humpback whale location (that represents the agent location in the search space) and the prey location (which represents the current best solution). Then, a spiral equation is performed between the whale location and the prey location to imitate the humpback whale on their spiral-shaped motion.

This behavior is modeled as following Eq. (11)⁵:

$$(11) \quad \vec{X}(t+1) = \vec{D}.e^{bl}.\cos(2\pi l) + \vec{X}.(t)$$

where D indicates the distance of the i^{th} whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in $[-1,1]$, and is an element-by-element multiplication.

Humpback whales swim around the prey within a shrinking circle and along a spiral-shaped path simultaneously. To model this simultaneous behaviour, we assume that there is a probability of 50% to choose between either the shrinking encircling mechanism or the spiral model to update the position of whales during optimization. The mathematical model is as follows¹⁷:

$$(12) \quad \vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A}.\vec{D} & \text{if } P < 0.5 \\ \vec{D}.e^{bl}.\cos(2\pi l) + \vec{X}^*(t) & \text{if } P \geq 0.5 \end{cases}$$

where p is a random number in $[0, 1]$.

In addition to the bubble-net method, humpback whales search for prey randomly. The mathematical model of the search is as follows:

where A plays an important role in order to guarantee the exploration phase of WOA, the other search agents update their locations based on the best solution when $A > 1$ which means the algorithm is still in the exploration phase.

The following Eq. (13) and Eq. (14) represent the mathematical model of this idea¹⁷:

$$(13) \quad \vec{D} = \left| \vec{C}.\overrightarrow{X_{rand}} - \vec{X} \right|$$

$$(14) \quad \vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}.\vec{D}$$

where $\overrightarrow{X_{rand}}$ is a random position vector (a random whale) is chosen from the current population.

Pseudo-code of the Whale optimization algorithm¹⁷

Initialize the whale population X_i ($i=1, 2, 3, \dots, n$)
 Calculate the fitness of each search agent
 X^* =the best search agent
While ($t < \text{maximum number of iterations}$)
 for each search agent
 Update a, A, C, l and p
 If1 ($p < 0.5$)
 If2 ($|A| < 1$)
 Update the position of the current search agent by Eq.(1)
 else if2 ($|A| \geq 1$)
 Select a random search agent (X_{rand})
 Update the position of the current search agent by Eq.(8)
 end if2
 else if ($p \geq 0.5$)
 Update the position of the current search agent by Eq.(5)
 end if1
 end for
 Check if any search agent goes beyond the search space and amend it
 Calculate the fitness of each search agent
 Update X^* if there is a better solution
 $t = t + 1$

end while

return X^*

5. Comparison and Discussion

The Algorithm proposed by Paolo and Francis¹¹ is Interactive GA using the Genetic Algorithm to prioritize the requirements. Results indicate that IGA converges and improves the performance of GA (without interaction) in a substantial way while keeping the user effort (number of elicited pairs) acceptable¹¹. The main goal of this proposed technique is to minimize the disagreement between prioritized requirement's order and different constraints that may be foisted with the requirements or that are arises recursively by the end-users during the process of requirement prioritization. When individual requirements having a high fitness (i.e., a low disagreement with the constraints) cannot be distinguished, since their fitness function evaluates to a plateau, user input is requested interactively, to make the fitness function landscape better suited for further minimization. The prioritization process terminates when a low disagreement is reached, the time out is

reached or the allocated elicitation budget is over¹¹. Overall, the conclusion of this approach shows that IGA is more suitable and appropriate when scanty ranking attributes are connected with the requirements collected by the software analysts but the performance of the algorithm degrades at an increasing rate of error. IGA (Proposed algorithm) outperforms GA and AHP. The Algorithm Proposed by Masadeh et.al.³ in 2018 is RP-GWO uses the Grey Wolf algorithm to prioritize the requirements. In this technique, clusters are created and calculated. For this K-means clustering technique is used, after this centroid is calculated to find the Euclidean distance between the randomly chosen centroid and assigned wolf and this distance is necessary to calculate the fitness function. To prioritize the requirements of each cluster ratio is calculated to cost and value.

For requirement prioritization, each and every cluster has three best solutions that are α , β , and δ ; α - requirement that has the first local minimum ratio, β - second local minimum ratio and δ - third local minimum ratio and it continues until all cluster will be empty and then the three top solutions in every cluster will compare with other cluster's top solutions. Whereas; the requirement which has a minimum ratio comparing with other ratios will be prioritized as the most significant and drop it from its cluster and the search space³.

The result of this proposed algorithm shows that it performs best for a smaller number of clusters hence it degraded with the increasing number of clusters. This technique compared with AHP (Analytical Hierarchy Process) and found more efficient in terms of running time. This technique (RP-GWO) is far better than AHP by approximately 30%³.

The Algorithm Proposed by Masadeh et.al.¹² in 2018 is RP-WOA uses the Whale algorithm to prioritize the requirements.

In RP-WOA technique, a cluster is created, and distance is calculated between the randomly chosen whale and the assigned whale for each cluster. Then, fitness is calculated to assess the cluster. In the bestowed work, the weight of the given requirements is denoted as the ratio of cost overvalue that obtained from the given requirements. The requirement with minimum ratio in each of the cluster will be chosen as the local min ratio requirement from that cluster. The requirement that has a minimum ratio among the set of local min ratio requirements will be prioritized as the most important one and remove from the search space¹⁸. This process is continued until all clusters going to be empty. This algorithm was tested against the AHP (Analytical Hierarchy Process) and the result shows that the RP-WOA performs better than the AHP by nearly 40%. RP-WOA has better performance than AHP. In case of large dataset size this algorithm performed better than the AHP.

5. Conclusion

This paper has supplied a comparison and literature review of metaheuristic algorithms. A cursory delineation of each method is conferred in conjunction with their pseudo-code. A comparative examination is performed based on their performance in the respective research paper. To get useful comparisons of metaheuristic techniques, distinct statistical assessments can be conducted to analyze and examine the techniques. This study concludes that the metaheuristic algorithms are newly developed fast, efficient, work best on multi-objective problems, and outperform the traditional method. In the future, these metaheuristic algorithms must be analyzed or statistically compared to each other to get the efficient technique and many other population-based metaheuristic algorithms must be applied to prioritize the requirements.

References

1. Gartner Inc, Gartner identifies the Top 10 strategic technologies for 2011, Online Available: <http://www.gartner.com>.
2. David G Rosado, Eduardo Fernández-Medina and Eduardo B Fernandez, An analysis of security issues for cloud computing, *Journal of Internet Services and Applications* 4, 5 (2013). <https://doi.org/10.1186/1869-0238-4-5>
3. R. Masadeh, A. Alzaqebah, and A. Hudaib, Grey Wolf Algorithm for Requirements Prioritization, *Modern Applied Science*, 12(2), (January 2018) 54.
4. I. Sommerville, & P. Sawye, Requirements engineering, *A good practice guide*, John Wiley & Sons, 1997. ISBN: 978-0-471-97444-4.
5. D. Greer & D.W. Bustard, SERUM-Software engineering risk: Understanding and management, *The International Journal of Project & Business Risk*, 1(4), (1997) 373-388.
6. P. Kousalya Reddy, S. G. M. Supraja & V.S. Prasad, Analytical Hierarchy Process approach—An application of engineering education, *Mathematica Aeterna*, 10, (2012) 861.
7. M. Vestola, A comparison of nine basic techniques for requirements prioritization, *Helsinki University of Technology*. M. Young, *The Technical Writer's Handbook*. Mill Valley, CA: University Science, (2010).
8. J. Karlsson, C. Wohlin and B. Regnell, An evaluation of methods for prioritizing software requirements, *Information and Software Technology*, vol. 39, pp. 14-15, (1998) 939-947.

9. A. Hudaib, R.Masadeh, M.H. Qasem, & A. Alzaqebah, Requirements Prioritization Techniques Comparison, *Modern Applied Science*, 12(2), (January 2018) 62.
10. P. Berander, and A. Andrews, Requirements prioritization, *Engineering and Managing Software Requirements*, Springer Berlin Heidelberg, (2005) 69-94.
11. Paolo Tonella, Angelo Susi, Francis Palma, Interactive Requirements Prioritization using a Genetic algorithm, *Information and Software Technology*, DOI: <http://dx.doi.org/10.1016/j.infsof.2012.07.003>. (January 2013).
12. R. Masadeh, A. Alzaqebah, and A. Hudaib, Whale Algorithm for Requirements Prioritization, *ICICS*, (April 2018).
13. Xin-She Yang, *Nature-Inspired Optimization Algorithms, Edition: 1st Edition; Publisher: Elsevier. Middlesex University, UK.* (March 2014).
14. Emad Elbeltagi, Tarek Hegazy, Donald Grierson, Comparison among five evolutionary-based optimization algorithms, www.elsevier.com/locate/aei. (October 2004)
15. Altaf Badar, Dr. B.S. Umre, Dr. A. S. Junghare, Study of Artificial Intelligence Optimization Techniques applied to Active Power Loss Minimization, *IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)* e-ISSN: 2278-1676, p-ISSN: 2320-3331.
16. S. Mirjalili, S. M. Mirjalili, & Lewis, Grey wolf optimizer, *Advances in Engineering Software*, 69, (2014) 46-61.
17. S. Mirjalili, and A. Lewis, The whale optimization algorithm, *Advances in Engineering Software*, 51-67, (2016).
18. W. A. Watkins and W. E. Schevill, Aerial observation of feeding behavior in four baleen whales *Eubalaenaglacialis*, *Balaenoptera borealis*, *Megapteranovaeangliae*, and *Balaenopteraphysalus*, *Journal of Mammalogy*, 60(1), (1979) 155-163.