Al-Rafidain Journal of Computer Sciences and Mathematics (RJCSM)

www.csmj.uomosul.edu.iq

Solving Reliability-Redundancy Allocation Problems using Grasshopper Optimization Algorithm

Mafaz Emad Abid Salih[1] and Najla Akram Al-Saati[2]

[1,2]Department of Software Science, College of Computer Sciences and Mathematics, University of Mosul, Iraq
Email: mafaz.23csp12@student.uomosul.edu.iq[1] and dr.najla_alsaati@uomosul.edu.iq[2]

Article information

Article history:
Received
Revised
Accepted
Published

Keywords:
Software Reliability,
Redundancy Allocation Problems,
Swarm Intelligence,
Grasshopper Optimization.

Correspondence:
Mafaz Emad Abid Salih
Email:
mafaz.23csp12@student.uomosul.edu.iq

Abstract

Success of companies and satisfaction of customers heavily depend on system reliability performance. System design improvements and higher efficiency exist directly from the proper distribution of reliability overhead resources. The extensive application domain of Reliability Redundancy Allocation Problems (RRAPs) includes fundamental challenges in various real-life situations such as software design along with cost optimization and development. The system reliability optimization problem is recognized as NP-Hard, and its resolution demands planned and effective solution approaches since there doesn't exist a polynomial-time method for finding optimal solutions. This research utilizes Grasshopper Optimization (GOA) due to its ability to solve complex constrained optimization problems effectively and its high accuracy in obtaining good solutions. Eight system reliability block diagrams were used, varying in their difficulty from simple to complex problems. Results showed that system reliability comprised a significant increase when GOA-based optimization was applied compared to other algorithms. GOA achieved higher performance using the eight system reliability block diagrams, and its results validated both the efficiency and the solution-delivering capabilities of the algorithm for enhancing overall software reliability.

DOI: 10.33899/rjcsm.v19i2.60316,

1. Introduction

The continuous progress of technology has led to significant industrial growth during the recent period. Production processes alongside systems now reach high complexity levels for attaining necessary productivity goals. The concept of system reliability has gained absolute importance in industrial operations. System efficiency during its operational lifetime demands this fundamental requirement due to modern high technology and escalating system complexity. The goal of reliability is to reach planned reliability standards in intricate systems by performing tests and verification together with planning processes. The redundancy and reliability allocation problem (RRAP) optimizes system reliability by managing cost limitations and weight and dimension requirements. Traditional optimization methods serve as the solution approach for these problems and utilize linear programming together with integer programming and fast gradient descent and sequential quadratic programming and gradient descent methods [1]. However, these methods exhibit significant limitations, including challenges in managing unexplored search spaces, a tendency to converge to single solutions, and weaknesses in local optimization processes [2]. Such shortcomings highlight the need for advanced intelligent optimization techniques to address the complexities inherent in real-life issues [3].

Software engineering relies heavily on the reliable nature of processes and products that include company-based research along with interests and decision-making activities. Reliability determines both customer choices and company achievement levels in business operations. According to [4] reliability refers to the statistical measure of how likely a part performs its requirements under predetermined situations over a defined timespan. The real-world problems which link closely with redundancy form an essential part of reliability studies. A subsystem's reliability improves when the quantity of redundant components grows directly impacting system reliability at the same time. This concept justifies adopting low-cost component designs with redundant backup systems although it reduces

overall reliability levels. RBD results when the composition of RBD system components reaches a complicated stage. Reliability control problems can be analyzed through mathematical modeling which enables selection of suitable approaches to their solution. Different real-world problems exhibit diverse characteristics which fall into two main groups as static or dynamic and linked or separate systems and single-objective or multi-objective requirements together with constrained or unconstrained status. Complexity alongside the numerous challenges in problem domains leads to poor performance rates when using traditional optimization algorithms [5].

Considering these challenges, Metaheuristic algorithms have received significant attention in the evolutionary computing community due to their remarkable speed, accuracy, and relatively low computational complexity. These algorithms have proven effective in addressing complex optimization problems across a variety of application domains [6].

Metaheuristic algorithms include evolutionary-based algorithms, inspired by natural evolutionary processes, such as Swarm Intelligence, tabu search, simulated annealing and others. A particularly interesting branch is swarm-based algorithms, inspired by swarming behavior observed in nature, which reproduce the self-organizing drifts of social individuals. A prominent example is the artificial bee colony (ABC) method [7], which mimics the foraging behavior of honeybee colonies.

Swarm algorithms have been used to find solutions to various types of problems in many diverse fields of science and technology. Some of them have been used to solve software reliability modeling [8], detection of diabetic diseases [9]. Moreover, Al-Isawi and Al-Saati also addressed swarm intelligence in finding the best control strategies to reduce risk management using the Spotted Hyenas optimization Algorithm [10].

The Grasshopper Optimization Algorithm (GOA) is a nature inspired metaheuristic, first presented by Saremi et al. (2017), which describes swarm behavior of grasshoppers in their native settings. Recently, GOA has experienced renewed interest and broad improvement making it one of the competitive swarm-based algorithms in optimization research [11]. GOA mimics both the long-range social interaction and the short-range repulsion in the grasshoppers producing a natural balance of finding the solution space between exploration, and exploitation.

One of the major advantages of GOA relates to the absence in its mechanism of the use of derivatives, which enables it to solve linear and non-linear, convex and non-convex tasks of continuous and discrete nature. In addition, its adaptive behavior to converge allows it to dynamically transform from wide exploration to fine exploitation as it approaches the best solution [12].

GOA has exhibited promising outcomes in various challenging engineering issues including multi-objective reliability optimization, feature selection, energy efficient scheduling and design of fault-tolerant systems. Modern variants (multi-objective GOA (MOGOA) and hybrid GOA structures (such as GOA-GA, GOA-PSO), have demonstrated success in solving NP-hard problems to improve robustness and generalization speed [13]. The ability of the algorithm to keep away from premature convergence and sustain population diversity also improves its suitability for reliability redundancy allocation problems (RRAP) and other real world optimization problems which need cost effective and robust system design, In addition, there are several applications of the hybrid algorithms in various software engineering fields such as classification [14] , medicine [15] even in planning a model to guide the establishment of a long-term human habitat on the Moon [16].

The main contribution of this study is to investigate the use of the GOA model, given its efficiency and robustness, in solving the RRAP problem, seeking to improve the overall reliability of systems under specific constraints. This research will be conducted using eight Reliability Block Diagrams (RBDs) of varying size and complexity.

The remainder of this paper is presented as follows: First, the problem statement is explained, followed by a detailed discussion of the criteria used to improve reliability in previous studies. Next, the concept of Grasshopper Optimization (GOA) is presented, along with its algorithm, and the results and comparisons obtained are presented. Finally, a conclusion is given regarding the proposed work and future work.

2. Related Work

Many researchers have used different approaches and techniques to solve RRAP problems. In 2018, Ghambari and Rahati established an improved artificial bee colony algorithm for reliability optimization problems through implementing a new search operator, they made available an improved user interface which led to better solutions for the reliability optimization problem [17] The authors Filho and Bessani introduced a linear multi-objective framework for RRAP in 2021 [18]. Al-Saati employed the wild horse algorithm for RRAP in 2022 resulting in superior maximum reliability than earlier works [19]. Kundu et al. introduced advances in hybrid algorithms through their work in 2022, they employed fitness metrics within their Hybrid Salp Swarm Algorithm with Teaching-Learning Based Optimization (HSSATLBO) method to show its effectiveness on a combination of benchmark problems [20]. In 2022, significant improvements were made to the optimization approach, as well as the methodology used by Yeh et al., which recorded several key metrics using the BAT-SSOA3 algorithm, Yeh et al. found results indicating that combining two algorithms consisting of a binary addition tree and small-sampled orthogonal tri-objective matrices produces results on strategic connectivity within a single integrated framework [21]. Maintaining consistency in this field, Khalil and Saleh in 2024 demonstrated the use of machine learning methods in estimating and predicting software reliability [22].

Krishna used serial and parallel redundancies to increase the dependability of cloud systems and services in 2024 using an Improved Modified Harmony Search (IMHS) combined with Modified Differential Evolution (MDE) [23], recently, Choudhary et al. used the Mayfly algorithm to solve the nonlinear mixed integer RRAP [24]. Also, Choudhary et al. considered the Cuckoo Search Algorithm (CSA) for solving the RRAPs in their study [25].

3. Mathematical Formulation of Rbd Problems

High reliability approaches need improved methodologies because systems become progressively complex. Measuring different optimization algorithm efficiency requires evaluating Reliability Block Diagram (RBD) problems as part of their assessment. The RBD problems that serve as fundamental elements during reliability method development are analyzed in this section. The formula of the general form can be demonstrated as follows:

$$\begin{gathered} Max\ R_s = f(n,r), \\ s.t.\ G(n,r) \le U \\ n_i \in +ve\ integers, 0 \le r_i \le 1,\ 1 \le i \le m \end{gathered}$$

where r_i denotes the i^{th} subsystem's reliability and n_i is the number of components in the i^{th} subsystem. Global dependability is represented by R_s, while the model's restrictions utilizing the upper bound U are represented by G. On the other hand, m stands for the number of subsystems. To optimize the reliability of the entire system, the goal is to ascertain the number of components and their dependability in each subsystem. This falls within the category of mixed-option nonlinear constrained integer optimization problems.

The basic Series System (RBD1) benchmark problem is the first one being examined, in which the system reliability is equal to the product of component reliabilities. It is important to remember that this is a fundamental difficulty that leads to more complex dependability issues [26]. In this study, the formula for this group has been used as Eq. (1) [17].

$$\begin{gathered} Max\quad fitness(n,r) = \prod_{i=1}^{5} R_i = \prod_{i=1}^{5}(1-(1-r_i)^{n_i}) \\ s.t,\ G_1 = \sum_{i=1}^{5} n_i^2 v_i \le V. \\ G_2 = \sum_{i=1}^{5} \alpha_i \left(n_i + e^{\left(\frac{n_i}{4}\right)}\right)\left(\frac{-1000}{\ln(r_i)}\right)^{\beta_i} \le C. \qquad (1) \\ G_3 = \sum_{i=1}^{5} n_i w_i e^{\left(\frac{n_i}{4}\right)} \le W. \\ \forall i = 1,2,\dots,5;\ 1 \le n_i \le 5, 0 \le r_i \le 1, \qquad n_i \in Z^+. \end{gathered}$$

where the physical parameters for each subsystem are represented by the symbols $v_i,\ \alpha_i, \beta_i\ and\ w_i$.

A Series-Parallel System (RBD2), which combines elements of both parallel and series systems [27], is used in the second issue. Its formula is shown in Eq. (2):

$$\begin{gathered} Max\quad fitness(n,r) = 1-(1-R_1R_2)\left[1-\left(1-(1-R_3)(1-R_4)\right)R_5\right] \\ s.t,\ G_1 = \sum_{i=1}^{5} n_i^2 v_i \le V. \\ G_2 = \sum_{i=1}^{5} \alpha_i \left(n_i + e^{\left(\frac{n_i}{4}\right)}\right)\left(\frac{-1000}{\ln(r_i)}\right)^{\beta_i} \le C. \qquad (2) \\ G_3 = \sum_{i=1}^{5} n_i w_i e^{\left(\frac{n_i}{4}\right)} \le W. \\ \forall i = 1,2,\dots,5;\ 1 \le n_i \le 5, 0 \le r_i \le 1, \qquad n_i \in Z^+. \end{gathered}$$

Following this, the third problem, which is called Complex System (RBD3), presents a challenging complex challenge that requires advanced optimization due to its many pieces and possibilities. Examining the level of optimality in resolving intricate dependability issues might make this issue more significant [20]. The following formula was used:

$$\begin{gathered} Max\quad fitness(n,r) = R_1R_2 + R_3R_4 + R_1R_4R_5 + R_2R_3R_5 - R_1R_2R_3R_4 \\ -R_1R_2R_3R_5 - R_1R_2R_4R_5 - R_1R_3R_4R_5 - R_2R_3R_4R_5 + 2R_1R_2R_3R_4R_5 \\ s.t,\ G_1 = \sum_{i=1}^{5} n_i^2 v_i \le V. \\ G_2 = \sum_{i=1}^{5} \alpha_i \left(n_i + e^{\left(\frac{n_i}{4}\right)}\right)\left(\frac{-1000}{\ln(r_i)}\right)^{\beta_i} \le C. \qquad (3) \\ G_3 = \sum_{i=1}^{5} n_i w_i e^{\left(\frac{n_i}{4}\right)} \le W. \\ \forall i = 1,2,\dots,5;\ 1 \le n_i \le 5, 0 \le r_i \le 1, \qquad n_i \in Z^+. \end{gathered}$$

The fourth issue pertains to the Overspeed System (RBD4) in unique circumstances when operational characteristics impact dependability. Scalability is measured by this benchmark and is significant in real-world applications [20], the formula for it as below:

$$\begin{gathered} Max\quad fitness(n,r) = \prod_{i=1}^{m}[1-(1-r_i)^{n_i}] \\ s.t,\ G_1 = \sum_{i=1}^{m} n_i^2 v_i \le V. \\ G_2 = \sum_{i=1}^{m} \alpha_i \left(n_i + e^{\left(\frac{n_i}{4}\right)}\right)\left(\frac{-1000}{\ln(r_i)}\right)^{\beta_i} \le C. \qquad (4) \\ G_3 = \sum_{i=1}^{m} n_i w_i e^{\left(\frac{n_i}{4}\right)} \le W. \\ \forall i = 1,2,\dots,m;\ 1 \le n_i \le m, 0.5 \le r_i \le 1-10^{-6}, n_i \in Z^+. \end{gathered}$$

Conversely, the Convex Quadratic System (RBD5) allows for the employment of many optimization approaches that investigate convexity features since its reliability functions are likewise convex [17].

$$Max \quad fitness(n,r) = \prod_{i=1}^{10} 1-(1-r_i)^{n_i}$$

$$s.t, \quad G_j = \prod_{i=1}^{10}(n_i^2 a_{ji} + n_i C_{ji}) \le b_j \qquad (5)$$

$i = 1,2,\dots,10.\,, j = 1,2,3,4.\,, n_i\ is + ve\ integer, n_i \in [1,6].$

where $0.8 \le r_i \le 0.99$ is generated by the uniform distribution so as to $0 \le a_{ji}, C_{ji} \le 10$ given that a_{ji}, C_{ji} are $+ve\ integers$ and $b_j = [2.0 * 10^{13}, 3.1 * 10^{12}, 5.7 * 10^{13}, 9.3 * 10^{12}]$.

A Mixed Series-Parallel System (RBD6) is a problem with a complicated series and a parallel system. The behavior of the optimization methods under different settings can be captured by this benchmark problem [20]. Its formula is as follows:

$$Max \quad f(n,r) = \prod_{i=1}^{m}[1-(1-r_i)^{n_i}]$$

$$s.t, \quad h_1 = \sum_{i=1}^{m} n_i c_i \le 400$$

$$h_2 = \sum_{i=1}^{m} n_i w_i \le 414 \qquad (6)$$

$$0.5 < r_i < 1\,, n_i \in Z^+, n_i \ge 1\,, m = 15.$$

The Large-Scale System (RBD7) deals with the difficulties of maximizing the dependability of large systems with many different parts [28]. It may be proven to be as follows:

$$Max \quad fitness(n,r) = \prod_{i=1}^{m}[1-(1-r_i)^{n_i}]$$

$$h_1 = \sum_{i=1}^{m} n_i^2 \alpha_i - \left(1+\frac{\theta}{100}\right)\sum_{i=1}^{m} l_i^2 \alpha_i \le 0$$

$$h_2 = \sum_{i=1}^{m} e^{\left(\frac{n_i}{2}\right)}\beta_i - \left(1+\frac{\theta}{100}\right)\sum_{i=1}^{m} e^{\left(\frac{l_i}{2}\right)}\beta_i \le 0$$

$$h_3 = \sum_{i=1}^{m} n_i \gamma_i - \left(1+\frac{\theta}{100}\right)\sum_{i=1}^{m} l_i \gamma_i \le 0 \qquad (7)$$

$$h_4 = \sum_{i=1}^{m} \delta_i \sqrt{n_i} - \left(1+\frac{\theta}{100}\right)\sum_{i=1}^{m} \sqrt{l_i}\delta_i \le 0$$

$$1 \le n_i \le 10, i = 1,\dots,m\ where\ m = 50, \qquad n_i \in Z^+.$$

here l_i means the lower bound of n_i, usually $\theta = 0.33, l_i = n_i$, $0.95 \le r_i \le 1 - 10^{-8}$, $m = 50$.

$b_i = (543;\ 352;\ 1040;\ 2048)$,$n_i = 1\ \forall\ i$,but $n_i = 2$ when $i = (4;\ 10;\ 15;\ 21;\ 33;\ 42;\ 45)$

and $\forall\ i = 1,\dots,50$:

α_i =(8;10;10;6;7;10;9;9;7;6;6;10;9;10;7;10;10;8;10;7;6;6;7;8; 9;8;8;9;10;9;7;9;6;7;6;10;9;10;6;8;10;8;8;6;6;8;7;10;8;10).

β_i =(4;4;4;3;1;4;2;3;4;4;5;3;1;4;4;2;1;3;5;4;2;2;2;5;5;1;3;3;1; 2;5;5;3;3;5;5;5;5;2;3;5;3;1;4;4;1;4;2;3;2).

γ_i =(13;16;12;12;13;16;19;15;12;16;14;15;17;20;14;13;15;19; 18;13;15;12;20;19;15;18;16;15;18;19;15;11;15;14;15;17; 19;11;17;17;17;18;18;19;13;19;14;19;15;11).

δ_i =(26;32;23;24;26;31;38;29;23;31;28;30;34;39;28;25;29;38; 36;26;30;24;40;38;29;35;32;29;35;37;28;22;29;27;29;33; 37;22;34;33;33;35;35;38;26;37;28;37;30;22).

The Incomplete Fault Detecting System (RBD8), which analyzes the dependability of systems in which not all defects can be identified, is the last issue. It may be expressed [17] as:

$$Max \quad fitness(n) = \prod_{i=1}^{4} R_i(n_i)$$

$$s.t, \quad G_1 = \sum_{i=1}^{4} n_i^2 d_{1i} \le 100$$

$$G_2 = \sum_{i=1}^{4} \left(n_i + e^{\left(\frac{n_i}{4}\right)}\right) d_{2i} \le 150 \qquad (8)$$

$$G_3 = \sum_{i=1}^{4} \left(n_i + e^{\left(\frac{n_i}{4}\right)}\right) d_{3i} \le 160$$

Where $n_i \in [1,6]\ for\ i = 1,2,4\ but\ n_3 \in [1,5]$. $R_i(n_i)$ can be evaluated as the following

$$R_1(n_1) = 1 - q_1\left(\beta_1 + q_1(1-\beta_1)\right)^{n_1-1}$$

$$R_2(n_2) = 1 - \frac{q_2\beta_2 + q_2^{n_2} p_2 (1-\beta_2)^{n_2}}{p_2 + q_2\beta_2}$$

$$R_3(n_3) = 1 - q_3^{n_3} \qquad (9)$$

$$R_4(n_4) = 1 - q_4\left(\beta_4 + q_4(1-\beta_4)\right)^{n_4-1}$$

4. Biological Life of Grasshopper Optimization Algorithm (Goa)

Among hemimetabolous insects, the grasshopper group predominates in the Caelifera suborder **Figure 1** research on grasshoppers continues because they possess outstanding behavioral abilities that allow shifts between single and swarm phases. Grasshopper mobility remains limited together with their individual patterns when habitat density levels are low. The population expands during suitable environmental conditions which leads to polyphenic transformation that results in the emergence of social swarming groups across expansive

ranges [29]. Biology demonstrates that grasshopper swarms perform dynamic movements through visual communications that are enhanced by mechanical signals combined with chemical signals. The movement of these individuals is controlled by three basic factors. [30][31]

- Through social contact, grasshoppers can escape isolation or congestion by adjusting their position to maintain an ideal distance from neighbors.
- The gravitational pull imitates the inclination to gravitate toward the earth or below.
- The external environmental influence that changes their trajectory is represented by wind advection.

Figure 1. Grasshopper in nature.

The Grasshopper Optimization Algorithm (GOA) bases its operations on treating grasshopper swarms as solution populations while using balance dynamics to express exploration with wide food searches and exploitation with local food refinement. The mathematical models developed by GOA derive from entomological and biomechanical research studies for replicating attractive-repulsive systems.

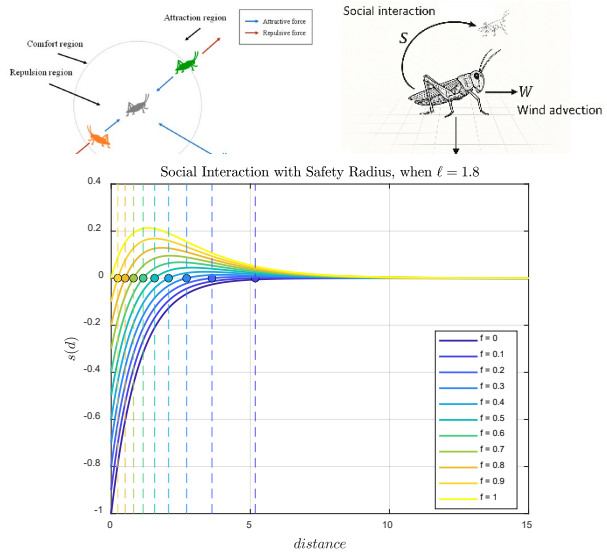
The grangerization process starts with neurophysiological responses to serotonin levels, according to recent biological studies that affect swarm dynamics. The phenomenon appears in GOA because this algorithm manages solution interaction strength [32]. Grasshopper swarms show complex biological structures that enable them to resist stagnation while remaining adaptable when faced with their environment, just as optimization algorithms need these qualities. The biological model of grasshopper motion provides numerous ideas for developing search heuristics that possess self-regulation and multi-modality and optimize the process by escaping local optima. The processes in GOA are ideally suited for complex optimization problems, including RRAP [33].

5. The Grasshopper Optimization Algorithm (Goa)

In 2017 Saremi along with Mirjalili and Lewis established the Grasshopper Optimization Algorithm (GOA) as a meta-heuristic inspired by nature. Mathematical modeling of grasshopper collective movement behavior throughout their life cycle allowed researchers to develop the GOA. Its attraction-repulsion behavior provides biologically valid coordination among individuals because it matches the natural movements grasshoppers use to stay spaced appropriately with peers and respond to environmental forces [31].

GOA uses each potential solution as a grasshopper that moves in a dimension space of D size. The three determining elements guide the movement of each grasshopper throughout space.

- Gravity force (G): Pulls each grasshopper toward the center of mass (often the current best solution) see **Figure 1**(a).
- Wind advection (W) enables stochastic exploration to maintain global search ability while promoting diversity across the search domain see **Figure 1**(a).
- The relationship between grasshoppers (S) utilizes exponential decay to calculate their overall attraction and repulsive forces shown in **Figure 2**(b).



(b)

Figure 2. (a)The effect of attractive and repulsive forces on a grasshopper's decision-making. (b) The social interaction distance, the small circles represent the comfort zone [31].

GOA presents a distinctive approach through flexible positional updates that enables an automatic transition between discovery-focused exploration and exploitation of best solutions without predefined guidelines [29]. Date from benchmark studies demonstrates GOA's excellent performance abilities both in restricted conditions and unrestricted scenarios. presented GOA as having powerful capabilities for global searching and fast convergence in addition to flexible mixed-integer optimization of redundancy allocation problems.

Sluggish movement and small steps are the main characteristics of the swarm at the larval stage. On the contrary, what sets the adult swarm apart is its sudden, long-range migration. The next important factor regarding grasshopper swarms is their search for food sources. Exploration and exploitation are the two logical dispositions into which the search process is divided for nature-inspired algorithms. The search agents would much rather travel locally when exploiting, but during exploration, they are encouraged to move quickly. Apart from target findings, the two tasks are also carried out automatically by grasshoppers [31].

The versatility of GOA has increased due to researchers developing multi-objective and hybrid frameworks extending from GOA-PSO and GOA-GWO that enable the efficient solution of complex RRAP cases which require cost and reliability and repeatability maintenance [33], GOA incorporates these three main characteristics:

- Biologically based swarm behavior.
- The optimization method adapts itself dynamically through uncomplex parameter control mechanisms.
- It effectively processes optimization problems having high dimensions, multiple restrictions and discrete components.

GOA has become a leading option to handle the complex nature and multi-modality found within the RRAP systems.

6. Goa Mathematical Framework

GOA is an algorithm developed out of the behavior of grasshoppers during exploration and exploitation. The principle states that when two operators are relatively close to one another (less than 2.079 units), they repel one another, and the operator is in the repulsion region or within the repulsion distance; when the two operators are exactly 2.079 units apart, there is neither an attractive nor a repulsive force; this is known as the comfort zone or the comfortable distance; conversely, when two operators are farther apart than 2.079 units, they attract one another; consequently, this distance is known as the attraction region or the distance of attraction [34].There is no absolute division between global and local optimization. When the process of increasing iteration times progresses, the size of the search region reduces, and the optimization process undergoes a change from large-scale global to refined localized processes. According to (10) [35].

$$p(d_{ij}) = \left[m\, e^{-\frac{d_{ij}}{n}} - e^{-d_{ij}} \right] \frac{x_j - x_i}{d_{ij}} \qquad (10)$$

The current grasshopper operator x_i is defined by the grasshopper operator x_j, where d_{ij} is the spatial separation between the current i^{th} and j^{th} grasshopper operators. The parameters m and n, which stand for the strength of attraction and the geographical scale of attraction, are used to assess how other agents affect the agent. The point to which the i^{th} grasshopper operator goes next in the k^{th} dimension is denoted by $\widehat{x_i^k}$ in (11), [35].

$$\widehat{x_i^k} = c_1 \left(\sum_{j=1}^{N} c_2 \frac{ul_k - fl_k}{2}\, p(d_{ij}^k) \frac{x_j^k - x_i^k}{d_{ij}^k} \right) + T_g^k \qquad (11)$$

which also defines the next position of the grasshopper operator x_i; The top and lower bounds of the agent in the k^{th} dimension are denoted by ul_k and fl_k, respectively; N is the total number of grasshopper operators; The current positions of the i^{th} and j^{th} grasshoppers in the k^{th} dimension are represented by x_i^k and x_j^k, respectively; d_{ij}^k indicates the spatial distance between the i^{th} and j^{th} grasshoppers in the k^{th} dimension; The adaptive shrinkage parameters c_1 and c_2, which preserve the relative balance between global and local optimization, are represented by T_g^k, which is the component of the best solution discovered thus far in the k^{th} dimension. The linear change of c_1 and c_2, denoted by c, may be computed as in (12) [35].

$$c = c_{max} - t\, \frac{c_{max} - c_{min}}{t_{max}} \qquad (12)$$

where t denotes a number of a current iteration, t_{max}notes a maximum number of iterations; c_{max} the largest of the adaptive shrinkage parameter values whereas c_{min} the adaptable shrinkage parameter minimum [35] The Main Phases upon which this algorithm is dependent on:

Phase 1: Initialization: Generate random locust locations within the search boundary.

Phase 2: Evaluation: Calculate the objective function for each locust.

Phase 3: An equation updates the locust positions.

The algorithm needs to be checked for both maximum iteration limits and when the best value of the fitness function stabilizes.

While the pseudo-code for this algorithm can be presented as:

Pseudo-Code: Grasshopper Optimization (GOA)

Place the initial values of population x_i where i ranges from 1 to N.
Evaluate fitness of each x_i
While ($t < Max\_iterations$)
  For each grasshopper x_i
    Calculate social interaction S
    Compute gravity G and wind W
    Update position x_i
 End For
 Evaluate fitness of updated Xi
Update global best solution
 $t = t + 1$
 End While
Return best solution found

7. The Parameters Employed in the Algorithm

In this section, an investigation is conducted to find values of the parameters that achieve the best possible results, in the context of finding better solutions to RBDs. Parameters were modified experimentally to suit the problem at hand, as in **Table 1**.

Table 1. Goa Parameters.

Parameter	Value
c_{max}	1
c_{min}	0.00004

8. Results And Comparisons

After applying GOA, results were compared with those of previous work to demonstrate the efficiency and validity of the used algorithm. The optimal fitness values indicate the maximum reliability of the system (typically ranging between 0 and 1), and these values show significant differences among algorithms across different studies in the comparison. The previous work considered in the comparisons are INGHS [36] and HSSATLBO [20]. The RBDs were grouped according to their complexity into simple RBDs and complex RBDs. **Table 2** provides a detailed comparison of GOA with other algorithms for the first four relatively simple RBD algorithms. **Table 3** shows the results of comparing GOA with other algorithms for the second four complex RBD algorithms. Some values of r_i and n_i were not available for some of the compared algorithms, making it impossible to recalculate and verify the acquired system reliability. These values are listed as unavailable in **Tables 2 and 3**.

Table 2. Comparing Reliability with Different Methods for Simple Problems.

RBD / Method	RBD1	RBD2	RBD3	RBD4
INGHS [36]	**Rs= 0.931682388** r_i= [0.7793988710, 0.8718370210, 0.9028853550, 0.7114025151, 0.787799488032] n_i = [3, 2, 2, 3, 3]	Rs= 0.999976649 r_i = [0.8198118626, 0.8449506842, 0.8956701585, 0.8952327069, 0.868438057445] n_i = [2, 2, 2, 2, 4]	Rs= 0.999889636 r_i = [0.8279847911, 0.8576796813, 0.9141564522, 0.6484814055, 0.7048654988] n_i = [3, 3, 2, 4, 1]	Rs= 0.9999546743 r_i = [0.9015565830, 0.8882438856, 0.9481110971, 0.8499817375] n_i = [5, 5, 4, 6]
HSSATLBO[20]	Rs= 0.931678 r_i= [N/A] n_i= [N/A]	Rs= 0.9999863372 r_i = [0.7753618512628, 0.8714241422773, 0.8903702230415, 0.8914438741116, 0.8630261550595] n_i= [3, 2, 2, 2, 4]	Rs= 0.9998896373815054 r_i = [N/A] n_i = [N/A]	Rs= 0.99995467466432 r_i= [0.901623877, 0.849936249, 0.948146758, 0.888204712] n_i= [5, 6, 4, 5]
GOA	Rs= 0.93168 r_i=[0.778902828997295 0.871855325826335 0.903169505800350 0.711475948663177 0.787719475235226] n_i= [3, 1, 4, 4, 5]	Rs= **0.999986337** r_i=[0.7715408, 0.863469348 0.89425877,3 0.893388044 0.865666241] n_i= [3, 2, 2, 2, 4]	Rs= **0.999966754** r_i=[0.82923432, 0.85822909, 0.913372226 0.64706684, 0.70431201] n_i= [3, 3, 2, 4, 1]	Rs= **0.999954675** r_i=[0.898515815 0.88561945,3 0.91540182 0.886067313] n_i= [5, 6,4, 5]

Table 3. Comparing System Reliability with Different Methods for Complex Problems.

RBD / Method	RBD5	RBD6	RBD7	RBD8
INGHS [36]	Rs=0.80884419 r_i =[N/A] n_i=[2,2,2,1,1,2,3,2,1,2]	Rs=0.945613358 r_i =[N/A] n_i =[3,4,6,4,3, 2,4,5,4,2,3,4,5,4,5]	Rs=0.40695475 r_i =[N/A] n_i =1 for all i except [4, 10, 15,21, 33,42,45] VTV	Rs=0.9745652160 r_i =[N/A] n_i =[3,3,2,3]
HSSATLB [20]	Rs=0.8088441896327347 r_i =[0.81, 0.93, 0.92, 0.96, 0.99, 0.89, 0.85, 0.83, 0.94, 0.92] n_i =[2, 2, 2, 1, 1, 2, 3, 2, 1, 2]	Rs=0.9456133574581371 r_i =[0.90 0.75 0.65 0.80 0.85 0.93 0.78 0.66 0.78 0.91 0.79 0.77 0.67 0.79 0.67] n_i =[3, 4, 6, 4, 3, 2, 4, 5, 4, 2, 3, 4, 5, 4, 5]	Rs= 0.4069547451370713 r_i=[0.995,0.974,0.965,0.971,0.968, 0.997,0.98,0.982,0.996,0.962, 0.972,0.979,0.961,0.987,0.962, 0.963,0.979,0.977,0.973,0.972, 0.97,0.973,0.982,0.987,0.994, 0.971,0.978,0.983,0.998,0.969, 0.979,0.977,0.97,0.974,0.991, 0.981,0.995,0.981,0.998,0.985, 0.977,0.96,0.988,0.974,0.962, 0.985,0.964,0.968,0.962,0.987] n_i =1 for all i except i=[4, 10, 15, 21, 33, 42, 45] is VTV	-
GOA	**Rs= 0.991075959** r_i =[10 of 0.99] n_i =[6, 6, 6, 6, 6, 6, 6, 6, 6, 6]	**Rs=0.99999688** r_i =[15 of 0.99] n_i =[4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 4]	**Rs=0.802192803** r_i =[50 of 0.97938 value] n_i =1 for all i except i= [4, 10, 15, 21, 33, 42, 45] is VTV	**Rs=0.9745652164** r_i=[2.903881207164253 2.881642360003297 2.295041021864859 3.175162277797285] n_i =[3 3 2 3]

Note: VTV (Variable taking value 2) is a position sequence of vector elements of the variables which take the value = 2 in the optimal case, while all the other variables take in the optimal case, the value = 1

As shown in **Table 2**, the results obtained are highly comparable to those of previous studies (shown in bold face) for 3 out of 4 RBD systems in terms of maximum achieved reliability, this clearly demonstrates its high performance combined with its extreme flexibility due to the influential parameter values applied to solve these problems.
The results in **Table 3** validate the effectiveness of GOA in solving larger and more complex RBD problems, achieving very high reliability values for all RBD problems compared to all other algorithms, with the best results indicated in bold face. The two tables demonstrate the outstanding performance of GOA in achieving very high reliability values, it achieves the best reliability for all four problems despite the problems being associated with more complex RBD problems, providing important insights into the high performance and efficiency of GOA's application to RRAPs.

Conclusion and Future Work

This paper has conducted an examination of RRAP through swarm intelligent methods to maximize system reliability with RBDs encompassing both parallel and series system designs. RRAPs belong to the class of NP-Hard problems, they seek to enhance reliability by meeting cost and weight, and volume restrictions. GOA was employed as a suitable solution method because of its recognized efficiency and effectiveness for this problem. Eight RBDs served as test models in this research, where systems consisted of simple, small designs in addition to complex and large ones. System reliability significantly increased based on results from this algorithm, as it proved to be very successful when compared with other methods in delivering results while managing exploration and exploitation phases. Although the GOA algorithm is powerful in solving optimization problems, it can fall to a local optimum and has a somewhat slow convergence speed. Future research should focus on finding solutions to such problems. In addition, this work has focused on homogeneous replication types only, future work may consider heterogeneous replication systems. A variety of swarm algorithms should be researched to demonstrate their effectiveness at providing better solutions for this problem class.

Conflict of interest

None.

References

[1] G. Li, T. Zhang, C.-Y. Tsai, L. Yao, Y. Lu, and J. Tang, "Review of the metaheuristic algorithms in applications: Visual analysis based on bibliometrics (1994–2023)," Expert Syst. Appl., p. 124857, 2024.

[2] A. Fadhil and T. Alreffaee, "Maintainability Prediction for Object-Oriented Software Systems Based on Intelligent Techniques: Literature Review," AL-Rafidain J. Comput. Sci. Math., vol. 14, no. 2, pp. 97–111, 2020.

[3] H. Fan, C. Wang, and S. Li, "Novel method for reliability optimization design based on rough set theory and hybrid surrogate model," Comput. Methods Appl. Mech. Eng., vol. 429, Art. no. 117170, 2024.

[4] G. J. Krishna, "Serial parallel reliability redundancy allocation optimization for energy-efficient and fault-tolerant cloud computing," arXiv preprint arXiv:2404.03665, Feb. 2024.

[5] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges," Artif. Intell. Rev., vol. 56, no. 11, pp. 13187–13257, 2023.

[6] A. Darwish, A. E. Hassanien, and S. Das, "A survey of swarm and evolutionary computing approaches for deep learning," Artif. Intell. Rev., vol. 53, no. 3, pp. 1767–1812, 2020.

[7] A. Karaman et al., "Hyper-parameter optimization of deep learning architectures using artificial bee colony (ABC) algorithm for high-performance real-time automatic colorectal cancer (CRC) polyp detection," Appl. Intell., vol. 53, no. 12, pp. 15603–15620, 2023.

[8] N. Al-Saati and M. Alabajee, "On the performance of Firefly Algorithm in software reliability modeling," Int. J. Recent Res. Rev., vol. 9, no. 4, pp. – , Dec. 2016.

[9] R. W. Ghanime and A. W. Ali, "Early detection of diabetic retinopathy using ResNet50," in Proc. 8th Int. Conf. Contemp. Inf. Technol. Math. (ICCITM), Mosul, Iraq, 2022, pp. 135–139.

[10] Z. Al-Isawi and N. Al-Saati, "Selecting the best control strategies for risk management using swarm intelligence," in Proc. Al-Sadiq Int. Conf. Commun. Inf. Technol. (AICCIT), 2023, pp. 310–313.

[11] L. Wu, J. Wu, and T. Wang, "Enhancing Grasshopper Optimization Algorithm (GOA) with Lévy flight for engineering applications," Sci. Rep., vol. 13, no. 1, Art. no. 124, 2023.

[12] H. Alirezapour, N. Mansouri, and B. M. Hasani Zade, "A comprehensive survey on feature selection with Grasshopper Optimization Algorithm," Neural Process. Lett., vol. 56, no. 1, Art. no. 28, Feb. 2024.

[13] V. H. S. Pham, V. N. Nguyen, and N. T. N. Dang, "Novel hybrid swarm intelligence algorithm for solving the capacitated vehicle routing problem efficiently," Evol. Intell., vol. 18, no. 3, pp. 1–26, 2025.

[14] N. Saleem and R. Saeed, "Classification software engineering documents based on hybrid model," Al-Rafidain J. Comput. Sci. Math., vol. 12, no. 2, pp. 61–87, 2018.

[15] O. Shakir and I. Saleh, "Hybridization of swarm for features selection to modeling heart attack data," Al-Rafidain J. Comput. Sci. Math., vol. 16, no. 2, pp. 25–34, 2022.

[16] T. Jin and A. Abedin, "Co-allocation of reliability–redundancy, spare parts and spacecraft trajectory for lunar habitation," J. Reliab. Sci. Eng., vol. 1, no. 1, Art. no. 015005, 2025.

[17] S. Ghambari and A. Rahati, "An improved Artificial Bee Colony algorithm and its application to reliability optimization problems," Appl. Soft Comput., vol. 62, pp. 736–767, 2018.

[18] D. de Quadros Maia Filho and M. Bessani, "Multi-objective linear approach for the reliability–redundancy allocation problem," in Proc. Int. Conf. Commun., Control Inf. Sci. (ICCISc), 2021, pp. 1–5.

[19] N. A. Al-Saati, "The exploration of Wild Horse Optimization in reliability-redundancy allocation problems," Int. J. Intell. Eng. Syst., vol. 15, no. 4, pp. 198–207, 2022.

[20] T. Kundu, Deepmala, and P. K. Jain, "A hybrid Salp Swarm Algorithm based on TLBO for reliability–redundancy allocation problems," Appl. Intell., vol. 52, no. 11, pp. 12630–12667, 2022.

[21] W. C. Yeh, W. Zhu, S. Y. Tan, G. G. Wang, and Y. H. Yeh, "Novel general active reliability redundancy allocation problems and algorithm," Reliab. Eng. Syst. Saf., vol. 218, Art. no. 108167, 2022.

[22] S. I. Khaleel and L. F. Salih, "A survey of predicting software reliability using machine learning methods," Int. J. Artif. Intell., vol. 13, no. 1, pp. 35–44, 2024.

[23] G. J. Krishna, “Serial parallel reliability redundancy allocation optimization for energy-efficient and fault-tolerant cloud computing,” arXiv preprint arXiv:2404.03665, 2024.

[24] J. Choudhary, A. S. Bhandari, and M. Ram, “Optimal system design for grid energy system using RRAP with MayFly optimization algorithm,” in Reliability Assessment and Optimization of Complex Systems. Elsevier, pp. 139–161, 2025.

[25] S. Choudhary et al., “Reliability optimization of off-grid solar power systems in households using Cuckoo Search algorithm,” in Reliability Assessment and Optimization of Complex Systems. Elsevier, 2025, pp. 179–190.

[26] M. Gen and Y. Yun, “Soft computing approach for reliability optimization: state-of-the-art survey,” Reliab. Eng. Syst. Saf., vol. 91, no. 9, pp. 1008–1026, 2006.

[27] M. Sharifi, A. Sayyad, S. Taghipour, and A. Abhari, “Optimizing a joint reliability–redundancy allocation problem with common-cause multi-state failures using immune algorithm,” Proc. Inst. Mech. Eng., Part O: J. Risk Reliab., vol. 237, no. 1, pp. 152–165, 2023.

[28] A. Bouaouda and Y. Sayouti, “Hybrid meta-heuristic algorithms for optimal sizing of hybrid renewable energy system: a review of the state-of-the-art,” Arch. Comput. Methods Eng., vol. 29, no. 6, pp. 4049–4083, 2022.

[29] S. J. Simpson and G. A. Sword, “Phase polyphenism in locusts: mechanisms, population consequences, adaptive significance and evolution,” in Phenotypic Plasticity of Insects: Mechanisms and Consequences, D. Whitman and T. N. Ananthakrishnan, Eds. Enfield, NH: Science Publishers, 2009.

[30] X. Ai, Y. Yue, and H. Xu, “Grasshopper optimization algorithms for parameter extraction of solid oxide fuel cells,” Front. Energy Res., vol. 10, Art. no. 853991, 2022.

[31] S. Saremi, S. Mirjalili, and A. Lewis, “Grasshopper optimisation algorithm: theory and application,” Adv. Eng. Softw., vol. 105, pp. 30–47, 2017.

[32] S. Abbasi, M. Seifollahi, F. Mohammadi, S. Khedmati, and M. Kheiry, “A modified Grasshopper Optimization Algorithm combined with wavelet functions,” Türk Hidrolik Derg., vol. 8, no. 2, pp. 1–15, 2024.

[33] S. Choudhary, M. Ram, N. Goyal, and S. Saini, “Analysis of reliability and cost of complex systems with metaheuristic algorithms,” Econ. Ecol. Socium, vol. 8, no. 1, pp. 1–15, 2024.

[34] A. Saxena, “A comprehensive study of chaos-embedded bridging mechanisms and crossover operators for Grasshopper Optimisation Algorithm,” Expert Syst. Appl., vol. 132, pp. 166–188, 2019.

[35] Y. Yang, W. Sun, and G. Su, “A novel support-vector-machine-based Grasshopper Optimization Algorithm for structural reliability analysis,” Buildings, vol. 12, no. 6, Art. no. 855, 2022.

[36] C. J. Kendall, M. Z. Virani, J. G. C. Hopcraft, K. L. Bildstein, and D. I. Rubenstein, “African vultures don’t follow migratory herds: scavenger habitat use is not mediated by prey abundance,” PLoS One, vol. 9, no. 1, Art. no. e83470, 2014.