International Journal of Swarm Intelligence and Evolutionary Computation

Neve et al., Int J Swarm Intel Evol Comput 2017, 6:3
DOI: 10.4172/2090-4908.1000165

Research Article Open Access

Application of Grasshopper Optimization Algorithm for Constrained and Unconstrained Test Functions

Abhishek G Neve*, Ganesh M Kakandikar and Omkar Kulkarni

Department of Mechanical Engineering, MIT, India

Abstract

Grasshopper Optimization algorithm is one of the recent algorithm for optimization. This algorithm is swarm based nature inspired algorithm which mimics and mathematically models the behaviour of grasshopper swarm in nature. The proposed algorithm can be used for solving the engineering optimization problems. The GOA is tested for different benchmark test functions to validate and verify the performance of the algorithm. Results obtained from GOA are compared with actual values (results) of the test functions. The results obtained from algorithm show that the algorithm is able to give the accurate results. The unconstrained and constrained test functions solved by using the Grasshopper optimization Algorithm (GOA) and the results can validate that the algorithm gives the trustable results. Constraints handling technique is used to convert the constrained optimization problem into unconstrained optimization problem, so that the problem can be handled by the Grasshopper Optimization Algorithm (GOA). Static penalty method is used as a constraints handling technique in this paper. The algorithm can also apply for different engineering problems in real life.

Keywords: Optimization; Grasshopper optimization algorithm; Constrained optimization; Swarm intelligence; Heuristic algorithm; Constrained and unconstrained test functions

Introduction

Simplest definition for optimization is "doing the most with the least". Lockhart and Johnson define optimization as "the process of finding the most effective or condition or favourable value". The aim of optimization is to achieve the "best" design relative to a set of prioritized criteria or constraints [1].

Different steps needs to be used to solve the optimization problem. The parameters and constraints of the problem should be identified. Based on parameters the problem may be differentiated as discrete or continuous and based on constraints the problem can be divided into constrained problem and unconstrained problem. Finally the optimization problem can be classified into single objective and multi objective problems depending on the nature of the objective function of the problem [2,3].

In mechanical design, Optimization process is a part of design in which some objectives such as weight, strength, deflection, wear etc. should be consider as per requirements. It is complicated to optimize the complete mechanical assembly as number of design variables is increased leads to complicated objective function. So optimization of individual parts or intermediate assemblies is much better and easy than the optimization of complete assembly. The aim in the optimization of a design is to minimize or maximize a design objective which satisfies the set of a given constraint for the design problem. In engineering design problem the design variables are usually of discrete or continuous type [4,5].

Stochastic optimization is a type of optimization. Stochastic methods use random operators and rely on them to avoid the local optima. In this method, one or a set of random solutions are generated at the beginning of optimization process for a given problem. In mathematical optimization it is necessary to calculate the gradient of the solution while in stochastic optimization only objective function(s) is required to find the solutions without need of gradient of solution. The problem is called as a black box as the decisions to improvise the solution is dependent on the objective function which is calculated. This is helpful in solving real life problems in which the search space is unknown. These advantages make the stochastic optimization popular over two decades [6].

Nature inspired swarm based algorithms are most popular among stochastic optimization approaches. Creatures in nature uses different techniques, which are used in such optimization techniques. The main aim of all creatures in nature is to survive and to achieve this goal they intend to evolve and modify as well as adapt different ways. So nature is the best inspiration as it is the best optimizer on the planet. These algorithms are of two types: (a) Single solution based, (b) Multi solution based. In single solution based type a single random solution is generated and improvised further while in multi solution based type multiple solutions are generated and modified. Usually multi solution based algorithms are chosen over single solution based algorithm [7,8].

The popular single-solution-based algorithms are hill climbing and simulated annealing. Other recent single-solution-based algorithms are Iterated Local Search (ILS) and Tabu Search (TS). Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Differential Evolution (DE) are different popular multi-solutions-based algorithms. The GA algorithm was inspired by the theory of "Survival of the fittest" proposed by Darwin in evolution. In this algorithm, parameters of the solutions are considered as the genes where the solution represents the individual. Poor solutions are continuously improved on the theory of Survival of fittest. The PSO algorithm is inspired from the foraging of schools of fishes or herds of birds. In this algorithm the best solution is the solution obtained by

***Corresponding author:** Abhishek G Neve, Research Scholar, Department of Mechanical Engineering, MIT, Pune, India, Tel: 02030273400; E-mail: neveabhi@gmail.com

Received August 07, 2016; **Accepted** August 31, 2017; **Published** September 06, 2017

Citation: Neve AG, Kakandikar GM , Kulkarni O (2017) Application of Grasshopper Optimization Algorithm for Constrained and Unconstrained Test Functions. Int J Swarm Intel Evol Comput 6: 165. doi: 10.4172/2090-4908.1000165

the particle and which is best among the swarm of particles. The Ant colony optimization (ACO) algorithm finds the best solution by using the collective behaviour of ants in finding the shortest path from the nest to the source of foods [9-17].

The similarity in both the types of nature inspired algorithm is that, the solution is improved until the end criterion is satisfied. The optimization process is divided in two phases for both types are exploration versus exploitation. In exploration as the algorithm possesses a tendency to have highly randomized behaviour and the solutions are changed significantly. Large changes in the solutions helps in searching of its promising regions because of greater exploration. In exploitation the solution face changes on smaller scale. After exploration the algorithm tends towards exploitation in which the algorithm is tend to search locally and on smaller scale. In a given optimization problem a proper balance between exploration and exploitation can result in finding the global optimum solution [18].

The literature shows various optimization algorithms formulated on the base of nature phenomenon. Due to their gradient-free mechanism, simplicity, high local optima avoidance, and considering problems as black boxes, in science and industry, nature inspired algorithms have been applied widely. Grasshopper optimization algorithm is based on the behaviour of the grasshopper when it searches its food and the same phenomenon is used in the algorithm to find the optimum solution for any optimization problem. The rest of the paper is organized as follows:

The grasshopper optimization algorithm is presented first, which contains the constraints handling technique for constrained optimization problem. Then the results are discussed on the optimization test functions and inspect the behaviour of the presented algorithm. Finally, conclusion of the work and suggestions in several directions for future studies are mentioned [19-21].

Grasshopper Optimization Algorithm (GOA)

Grasshoppers are insects and consider as a pest. They usually damage the crop production as well as agriculture which lead to consider them as pest. Usually we see the grasshopper individually in nature but most of the time they join large swarm among all creatures in nature. The swarm of grasshopper maybe a nightmare for farmers as the size of the swarm can be much large. The grasshopper swarm possesses one unique characteristic which is that we found the swarming behaviour in both the nymph as well as adulthood in grasshopper. The nymph grasshopper move like rolling cylinders in millions of numbers. They almost eat all the vegetation which comes in their path during their movement. When they become adult from nymph, they form a swarm in air and then they migrate over a very large distance [22,23].

The swarm usually has very slow movement when they are in larval phase. The small step of the grasshopper is main characteristic of the swarm in larval phase. Opposite of that the main feature of swarm in adulthood is long range and abrupt movement of swarm. Swarming of grasshopper is mainly formed for searching of food source. This food seeking of grasshopper is another characteristic of swarming of grasshopper. As discussed in introduction exploration and exploitation are two tendencies of nature inspired algorithm. Along with target seeking these both tendencies are performed by grasshoppers naturally in which they move abruptly as well as locally in small areas. A mathematical model is formed of this behaviour of grasshopper to design a nature inspired optimization algorithm.

The mathematical model employed to simulate the swarming behaviour of grasshoppers is presented as follows [24]:

$$X_i = S_i + G_i + A_i \tag{2.1}$$

Where X_i defines the position of the i^{th} grasshopper, S_i is the social interaction, G_i is the gravity force on the i^{th} grasshopper and A_i shows the wind advection. Note that to provide random behaviour, the equation can be written as

$$X_i = r_1 S_i + r_2 G_i + r_3 A_i$$

where r_1, r_2 and r_3 are random numbers in [0,1].

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(d_{ij}) \vec{d}_{ij} \tag{2.2}$$

Where d_{ij} is the distance between the i^{th} and the j^{th} grasshopper, calculated as,

$$d_{ij} = \left| x_j - x_i \right|,$$

s is a function to define the strength of social forces, as shown in Equation (2.3) and d_{ij} is a unit vector from the i^{th} grasshopper to the j^{th} grasshopper which can be defined as,

$$\vec{d}_{ij} = \frac{x_j - x_i}{d_{ij}}$$

The s function, which defines the social forces, is calculated as follows:

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \tag{2.3}$$

Where f indicates the intensity of attraction and l is the attractive length scale.

The function s shows the impacts on the social interaction (attraction and repulsion) of grasshopper.

The distances are considered from as 0 to 15. The interval of repulsion is (0, 2.079). The comfortable distance of a grasshopper is 2.079 units from other grasshopper, as there is neither attraction nor repulsion for a grasshopper when it is away from other grasshopper by 2.079 units. This is also called as comfortable zone.

For artificial grasshoppers there is difference in social behaviours as the parameters l and f in equation (2.3) changes. After varying l and f independently, the effects of these parameters on function s can be observed. The parameters l and f change comfort zone, attraction region and repulsion region effectively. It should be noted that the attraction or repulsion regions are very small for some values (l=1.0 or f=1.0 for instance). From all these values we have chosen l=1.5 and f=0.5 [25].

It may be pointed that, in simplified form, this social interaction was the motivating force in some earlier locust swarming models [26]. The space between two grasshoppers is divided into comfort zone, attraction region and repulsion region with the help of function s. With the distances greater than 10 the value of function returns the value close to zero. With large distances between grasshoppers, the strong forces cannot be applied by using this function. To overcome this problem the distance of grasshoppers is kept and mapped in the interval (1 4).

The G component in equation (2.1) is calculated as follows:

$$G_i = -g\vec{e}_g \tag{2.4}$$

Where g is the gravitational constant and e_g shows a unity vector towards the centre of earth.

The A component in equation (2.1) is calculated as follows:

$$A_i = u\vec{e}_w \tag{2.5}$$

Where u is a constant drift and e_w is a unity vector in the direction of wind. The movement of the nymph is highly correlated with the direction of wind as the nymph grasshoppers don't have wings. Substituting S, G and A in equation (2.1), this equation can be expanded as follows:

$$X_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} s(d_{ij})\vec{d}_{ij} - g\vec{e}_g + u\vec{e}_w \tag{2.6}$$

Where $s(r) = fe^{\frac{-r}{l}} - e^{-r}$ and N is the number of grasshoppers.

The position of nymph grasshoppers should be prevented from going below threshold when they land on ground. However this equation cannot be utilized in the swarm simulation and optimization algorithm as it prevents the algorithm from exploring and exploitation the search space around the solution. This model can be used in the free space for swarms only. The interaction between grasshoppers in swarm can be simulated by using equation 2.6.

The grasshopper reach the comfort zone very fast and swarm does not converge to a specified point and because of that the in optimization problem, the mathematical model cannot be used directly. To solve optimization problems, by doing some modifications a modified version of this equation is proposed as follows:

$$X_i^d = \left(\sum_{\substack{j=1 \\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}} \right) + \overline{T}_d \tag{2.7}$$

Where ub_d is the upper bound in the Dth dimension, lb_d is the lower bound in the Dth dimension $s(r) = fe^{\frac{-r}{l}} - e^{-r}$, $\overline{T}_d$ is the value of the Dth dimension in the target (best solution found so far) and C is a decreasing coefficient to shrink the comfort zone, repulsion zone, and attraction zone. Note that S is almost similar to the S component in equation (2.1).

However, we do not consider gravity (no G component) and assume that the wind direction (A component) is always towards a target $(\overline{T}_d)$.

The next position of grasshopper is shown by equation (2.7) is based on its current position, the position of target and the all other grasshoppers position. In this the first component which is considered as the location of current grasshopper and that is with respect to other grasshoppers. In fact, to define the location of search agents around the target, the status of all grasshoppers is considered. This is the main difference between GOA and PSO, which is considered as the most regarded swarm intelligent technique. In GOA only one position vector need to take into consideration while in PSO there are two vectors for each particle i.e. position and velocity vector. In PSO no other particle contributes in updating the position of a particle, whereas in GOA, all the search agents are required to define the next position of the each search agent.

It is also worth mentioning here that the adaptive parameter has been used twice in equation (2.7) for the following reasons:

- The first C from the left is same as inertial weight (w) used in PSO. The grasshopper movement is reduced around the target by it. This parameter is used to balance the exploration and exploitation of swarm around the target.

The second C helps to decrease the attraction zone, comfort zone, and repulsion zone between grasshoppers. The component $c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)$ in the equation (2.7), $c\frac{ub_d - lb_d}{2}$ linearly decreases the exploration and exploitation space by the grasshoppers. The grasshopper is either repelled from (explore) or attracted to (exploitation) and this is indicated by the component $s\left(\left|x_j^d - x_i^d\right|\right)$.

The inner c is responsible for reduction of repulsion/attraction forces between grasshoppers, which is proportional to the number of iterations, while the outer c is helpful in reducing the search coverage around the target as the iteration goes on increasing. In summary, in equation (2.7) the first term which is the sum is considers the position of other grasshoppers and accordingly implements the grasshopper interaction in nature.

The second term $\overline{T}_d$, simulates the tendency of grasshoppers to move towards the food source. The deceleration of grasshopper which is approaching towards the food source and eventually consumption of it is simulated by the parameter c. Both the terms in the equation is multiplied by the random values which provide more random behaviour to grasshoppers. An individual term in the equation is also multiplied by random numbers to provide random behaviour in both interactions of grasshoppers as well as their tendency towards the source of food.

The proposed mathematical formulations are able to explore and exploit the search space. To tune the level of exploration to exploitation for search agents, a mechanism is required. In nature the initial movement of grasshopper is locally for search of food as they are in larvae phase in which they have no wing. For larger region search they move freely in air to explore in their adulthood. However the exploration comes first in stochastic optimization algorithm as there is need for finding the promising region in the search space. After obtaining the promising region by exploration, local search is carried by exploitation to find the accurate approximation of the global optimum.

For balancing exploration and exploitation, there is relation between and number of iteration and accordingly the c is to be decreased. Exploitation is promoted as the number of iteration increases. The comfort zone is reduced by c proportional to iteration count and is calculated as follows:

$$c = c_{\max} - l\frac{c_{\max} - c_{\min}}{L} \tag{2.8}$$

Where c_{max} is the maximum value, c_{min} is the minimum value, l indicates the current iteration and L is the maximum number of iterations. In this work, we use l and 0.0001 for c_{max} and c_{min}, respectively.

From above it is seen that the mathematical model should help grasshopper to move towards a target gradually. In actual practice, the actual optimum solution is unknown to us which is exact global optimum. Therefore, it is required to find out the target for each grasshopper in every stage of optimization. In GOA, it is assumed that the grasshopper with best objective value is fittest grasshopper during optimization. This will help to save the best solution in each iteration in algorithm. This is done hope that the most accurate target is reached the approximation of real problem.

Constraint handling technique

To use the recent developed nature inspired, evolutionary algorithms, it is necessary to convert the constrained optimization problem into unconstrained optimization problem. Without loss of generality we may transform any optimization problem using constraint handling techniques. A variety of constraint-handling methods have been developed in the last decades. Penalty method is one of the constraints handling technique.

Penalty function: Penalty functions have been a part of the literature on constrained optimization for decades. This concept is a part of constraint handling technique. Constraint handling techniques are used to convert the constrained optimization problem into unconstrained optimization problem. The Grasshopper Optimization Algorithm is able to handle the unconstrained optimization problems only, so using the penalty function the constrained optimization problems can be solved using GOA.

Static penalty is used in most of the cases. In this approach a constant penalty is applied to those solutions which violate the feasibility of the solution. The general formulation of the penalty function is:

$$f_p(x) = f(x) \pm \sum_{i=1}^{n} r_i G_i + \sum_{j=1}^{p} c_j L_j \quad (2.9)$$

Where, G_i and L_j are functions of constraints $g_i(x)$ and $h_j(x)$ respectively [27].

$g_i(x)$ are inequality constraints, $h_j(x)$ are equality constraints, r_i and c_j are positive constants normally called "penalty factors".

$$G_i = \max\left[0, g_i(x)\right]^{\beta}$$

$$L_j = \left|h_j(x)\right|^{\gamma}$$

where β and γ are normally 1 or 2.

The pseudo code of GOA algorithm for unconstrained and constrained benchmark test problems is shown in Figures 1a and 1b. The random solutions are generated at very beginning of the algorithm. Based on equation (2.7) the positions of search agent are updated. In each iteration the best solution obtained is updated. The distances between grasshoppers are normalized in each iteration. Until the satisfaction end result is not obtained, the updating of position is continuously carried on. The position and fitness of the best target is finally returned as the best approximation for the global optimum (Figures 1a and 1b).

Results

This section presents the test problem with known optima and performance of GOA algorithm. The results are then provided and analyzed in details.

```
Initialize the swarm Xi (i = 1, 2, 3, ....., n)
Initialize cmax, cmin, and maximum number of
iterations Calculate the fitness of each search agent
T = the best search agent
while (l < Max number of iterations)
    update c using eq. (2.8)
    for each search agent
        Normalizes the distance between grasshoppers in [1,4]
        update the position of current search agent by the equation (2.7)
        bring the current search agent back if it goes outside the boundaries
    end for
    update T if there is a better solution
    l = l+1
End while
Return
```

Figure 1a: Pseudo code for unconstrained problem.

```
Initialize the swarm Xi (i = 1, 2, 3, ....., n)
Initialize cmax, cmin, and maximum number of
iterations Calculate the fitness of each search agent
T = the best search agent
while (l < Max number of
    iterations) update c using eq.
    (2.8)
    for each search agent
        Normalizes the distance between grasshoppers in [1,4]
        update the position of current search agent by the
        equation (2.7)
        bring the current search agent back if it goes outside
        the boundaries
    end for
    update T if there is a better
    solution l = l+1
End
while
Return
```

Figure 1b: Pseudo code for constrained problem.

Experimental setup

It is very common to employ mathematical test functions in stochastic optimization. The optima of the mathematical test functions are known, so the performance of the algorithm can be measured quantitatively [28].

It is required that the characteristics of the test function should be that diverse, so that they are able to reach to some proper conclusion. In this work, two sets of test functions are used to check the performance of the GOA algorithm. The test functions are constrained and unconstrained optimization functions. The mathematical test functions are available in appendix.

For solving the test functions, 500 iterations along with 30 search agents are employed. The algorithm was coded in Matlab (R2016a) on windows 7 platform with I5-3470 Processor 3.2 GHZ processor speed and 4 GB RAM. Each test was solved 30 times to generate the statistical results. The best solution among all results is chosen as final optimum solution. The qualitative results, along with trajectory of grasshopper, search history, convergence curves, and average fitness of population have been discussed and analyzed in following subsection.

Qualitative results and discussion

The experiment is conducted on both sets of test functions i.e. unconstrained and constrained test problems. The main objective behind conducting this experiment was to observe the GOA algorithm behaviour qualitatively. Four diagrams have been drawn for each of the test function shown in these diagrams (Figure 2):

- Average fitness: This diagram indicates the average objective value of all grasshoppers in each iteration.
- Convergence curve: This diagram shows the objective value of the best solutions obtained so far (target) in each iteration.

As shown in Figure 2, grasshoppers tend to explore the different regions of the search space around the global optima eventually. These results show that the GOA algorithm beneficially balances exploration and exploitation to drive the grasshoppers towards the global optimum.

The trajectory curves in Figure 2, show that the grasshoppers made abrupt changes in initial steps of optimization largely. Exploration

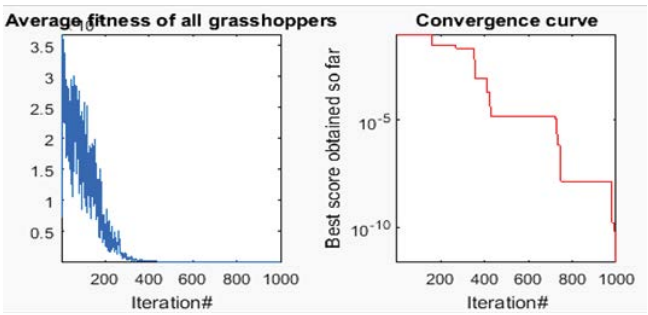

Figure 2a: Average fitness of all grasshoppers and convergence curve.

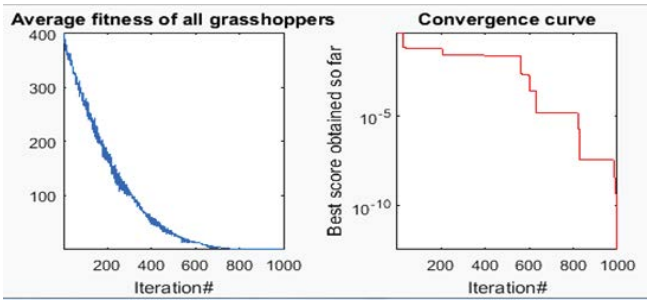

Figure 2b: Quantitative results of all grasshoppers and convergence curve.

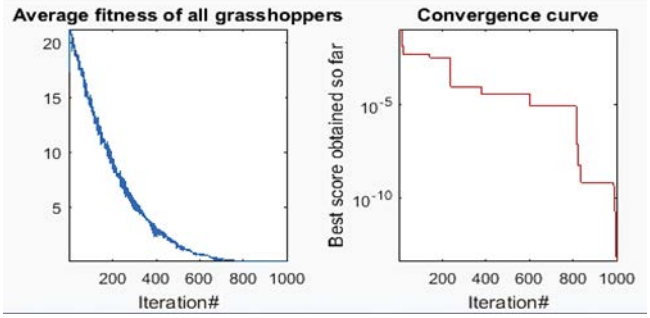

Figure 2c: Unconstrained test functions Curves.

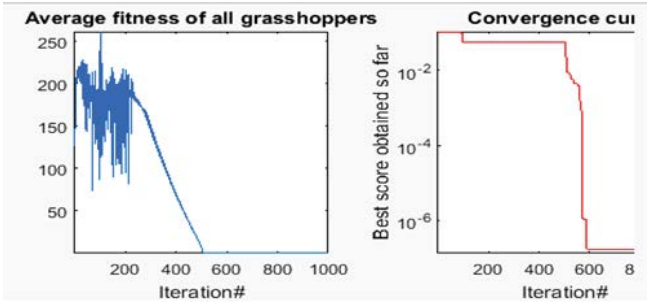

Figure 2d: Constrained test functions curves.

of search space is takes place due to high repulsive rate of GOA. It is also seen that, as the optimization approaches further the fluctuation decreased gradually. This is done due to the attraction forces as well as comfort zone between grasshoppers. This guarantees that the algorithm

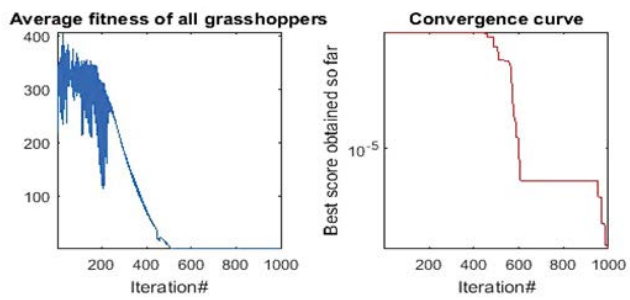

Figure 2e: Grasshoppers and convergence curves.

	Actual (Adorio EP) [28]			GOA		
Function	**x1**	**x2**	**f**	**x1**	**x2**	**f**
Beale's function	3	0.5	0	3	0.5	2.23E-12
Beale's function	1	3	0	1	3	3.72E-13
Matyas function	0	0	0	(-1.4191e-07)	2.47E-07	3.78E-14

Table 1: Results of GOA and comparison with actual values (unconstrained test functions).

	Actual (Adorio EP) [28]			GOA		
Function	**x1**	**x2**	**f**	**x1**	**x2**	**f**
Rosenbrock function constrained with a cubic and a line	1	1	0	0.99964	0.99927	1.32E-07
Rosenbrock function constrained to a disk	1	1	0	0.99996	0.99992	1.52E-09

Table 2: Results of GOA and comparison with actual values (constrained test functions).

will converge to a point eventually because of exploration and exploitation. The curve shows the descending behaviour and it proves that the GOA enhances the initial random population and improves the accuracy of the approximated optimum over the course of iterations.

Quantitative results and discussion

In above section, the qualitative results are discussed and demonstrated that GOA is able to solve the optimization problems. The test functions are of 2 variables with both constrained and unconstrained type problems. The experimental results are shown in Tables 1 and 2 for unconstrained and constrained test functions as well as the actual values of test functions are compared with GOA results.

In Table 1, the results of unconstrained test functions are compared. The test functions are solved by GOA and the solution obtained from this algorithm is compared with the actual results of the test functions and from there it is seen that the solution obtained by GOA is approximately same. From this we can say that the GOA is giving the accurate solution of optimization problems.

In Table 2, the results of constrained test functions are compared. To solve the constrained optimization problem in GOA, constraints handling technique is required. In here the penalty approach is used as a constraint handling technique. With the help of this technique the problem is converted into unconstrained problem and then solved it. The results obtained of this constrained test functions by GOA are compared with actual results of test functions. From comparison we can definitely say that the results given by GOA are very much near to actual solution and consider as the accurate solution of the problem.

Function	Objective function	Range	f_{min}
Beale's function	$(1.5-x+xy)^2+(2.25-x+xy^2)^2+(2.625-x+xy^3)^2$	[-4.5 to 4.5]	0
Beale's function	$(x+2y-7)^2+(2x+y-5)^2$	[-10 to 10]	0
Matyas function	$0.26(x^2+y^2)-0.48xy$	[-10 to 10]	0

Table 3: Unconstraint test functions.

Function	Objective function	Constraints	Range	fmin
Rosenbrock function constrained with a cubic and a line	$(1-x)^2+100(y-x^2)^2$	$(x-1)^3-y-1<0$ $x+y-2<0$	[-1.5 to 1.5] [-0.5 to 2.5]	0
Rosenbrock function constrained to a disk	$(1-x)^2+100(y-x^2)^2$	$x^2+y^2<2$	[-1.5 to 1.5] [-1.5 to 1.5]	0

Table 4: Unconstraint test functions.

From the above discussion it is clear that the results obtained by using GOA are very close to actual and accurate solutions. So GOA can be used efficiently in constrained as well as unconstrained optimization problems and we can rely on the solution or results obtained from GOA. The discussions and findings of the above section clearly state the quality of exploration, exploitation, local optima avoidance, of the algorithm. The repulsive for, attractive force between grasshoppers changes along with the increase in the iterations. All these parameters of the grasshoppers are useful in the algorithm to avoid the local optima of the optimization problem and to converge to the global optimum solution accurately and quickly (Tables 3 and 4).

Conclusion

This work presented the optimization algorithm called grasshopper optimization Algorithm used to validate the results of GOA by using optimization test functions. Both constrained and unconstrained optimization test functions are used to validate the results obtained from GOA. A mathematical model is studied which is based on the swarming behaviour of grasshopper in nature. A mathematical model simulates the repulsive and attractive forces between the grasshoppers. GOA contains a coefficient that adaptively decreases the comfort zone which is used in balancing of exploration and exploitation. Finally the best solution given by swarm is considered the optimum solution of the optimization problem.

In order to validate the performance of the GOA, the test functions are used in paper. The test functions are solved by the using Grasshopper optimization algorithm. The results obtained by GOA are observed to check the performance of the algorithm qualitatively and quantitatively. The experiment and discussion support the following conclusions:

- Grasshopper optimization algorithm avoids local optima and able to find the global optima in the given space.
- GOA balances exploration and exploitation to find the global optimum solution of optimization problem.
- Grasshoppers are able to find the most promising region from the given search space.
- For unconstrained optimization problem the GOA gives accurate solutions. The optimum point obtained from GOA is accurate and promising.
- The GOA also gives correct and promising results for constrained optimization problem.

GOA can be used only for single objective problems. For future work, the algorithm can be developed for multi objective problems. GOA can contribute into different real world optimization problems. Tuning the main controlling parameters of GOA may also be beneficial.

References

1. Kelley TR (2005) Optimization, an important stage of engineering design. Technol Teacher 69: 18-23.
2. Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput Meth Appl Mech Eng 191: 1245-1287.
3. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscipl Optim 26: 369-95.
4. Rao RV, Savsani VJ (2012) Mechanical design optimization using advanced optimization techniques, Springer Series in Advanced Manufacturing.
5. Deb K, Goyal M (2007) A flexible optimization procedure for mechanical component design based on genetic adaptive search. Mechanical Engineering department, Indian Institute of Technology, Kanpur.
6. Dasgupta D, Michalewicz Z (1997) Evolutionary algorithms in engineering applications. Springer.
7. Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver press.
8. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9: 1-14.
9. Davis L (1991) Bit-climbing, representational bias and test suite design. ICGA, pp: 18-23.
10. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220: 671-80.
11. Lourenço HR, Martin OC, Stutzle T (2001) Iterated local search. arXiv preprint math/0102188.
12. Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution.
13. Glover F (1989) Tabu search-part I. ORSA J Comput 1: 190-206.
14. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth International Symposium on Micro Machine and Human Science, pp: 39-43.
15. Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, pp: 134-142.
16. Holland JH (1992) Genetic algorithms. Sci Am 267: 66-72.
17. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11: 341-59.
18. Eiben AE, Schippers C (1998) An evolutionary exploration and exploitation. Fund Inform 35: 35- 50.
19. Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237: 82-117.
20. Gogna A, Tayal A (2013) Metaheuristics: review and application. J Exp Theor Artif Intel 25: 503-526.
21. Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN, et al. (2011) Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm Evol Comput 1: 32-49.
22. Simpson SJ, McCaffery A, Haegele BF (1999) A behavioural analysis of phase change in the desert locust. Biol Rev 74: 461-80.
23. Rogers SM, Matheson T, Despland E, Dodgson T, Burrows M, et al. (2003) Mechanosensory-induced behavioural gregarization in the desert locust *Schistocerca gregaria*. J Exp Biol 206: 3991-4002.
24. Topaz CM, Bernoff AJ, Logan S, Toolson W (2008) A model for rolling swarms of locusts. Eur Phys J Special Top 157: 93-109.
25. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: Theory and application. Adv Eng Soft 105: 30-47.
26. Lewis A (2009) Low Cost: a spatial social network algorithm for multi-objective optimisation. In: Evolutionary computation, CEC'09. IEEE congress, pp: 2866-2870.

27. Kulkarni O, Kulkarni N, Kulkarni AJ, Kakandikar G (2016) Constrained cohort intelligence using static and dynamic penalty function approach for mechanical components design. Int J Parallel Emergent Distrib Sys.

28. Adorio EP (2015) MVF-multivariate test functions library in c for unconstrained global optimization.