2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)
Dec. 22nd, 2017
(Iran University of Science and Technology) – Tehran, Iran

Spring Search Algorithm: A new meta-heuristic optimization algorithm inspired by Hooke's law

Mohammad Dehghani
PhD. Student
Dept. of Electrical and Electronics Engineering Shiraz University of Technology
Shiraz, I.R. Iran
adanbax@gmail.com

Zeinab Montazeri
M.Sc. Student
Dept. of Electrical Engineering Islamic Azad University of Marvdasht
Marvdasht, I.R. Iran
Email: Z.montazeri2017@gmail.com

Ali Dehghani
MSc. Student
Dept. of Civil Engineering Islamic Azad Universities of Estahban
Estahban, I.R. Iran
adanbax@yahoo.com

AliReza Seifi
Prof.
Dept. of Power and Control Shiraz University
Shiraz, I.R. Iran
seifi@shirazu.ac.ir

Abstract—**Nowadays, accident-based heuristic swarm algorithms are widely used for optimization. The important class of these algorithms has been developed by ideas of physical processes or behaviors of creatures. This paper introduced a new method to achieve semi-optimized solutions related to optimization problems in the various sciences. The proposed method of spring search algorithm is among the optimization algorithms developed by idea of laws of nature and the search factors are a set of objects. The paper presents the real version of the algorithm. The standard multi-function optimization results indicate efficiency and optimal performance of the proposed method.**

Keywords-heuristic algorithms; optimization; spring force optimality; spring.

I. INTRODUCTION

Due to the complexity of the issues and the importance of achieving to solution rapidly, other classical methods of optimization are not able to solve many of the problems anymore, and randomized search algorithms are mainly used instead of comprehensive search of problem statement space. This has led heuristic search algorithms to be used increasingly in recent years [1, 2, 3, 4, 5]. Heuristic algorithms have shown their high capabilities in many fields of science such as logistics, bioinformatics, data mining, chemical physics, electronics, and many other related areas. To achieve an appropriate mathematical model to search heuristic methods is very difficult task, and even impossible [8]. Therefore, these types of algorithms can be called as "black box" optimization algorithms [11]. In swarm methods, interactions and exchange of information between members are performed in different ways. The examples of these algorithms include genetic algorithm inspired by genetics and evolution (1975), simulated annealing inspired by observations of thermodynamics (1983), artificial immune by simulation of human defensive system (1986), ant colony algorithm by simulating the behavior of ants in search of food (1991), and particle swarm optimization by simulating the social behavior of birds (1995) [1, 2, 3, 4 and 5]. In this paper, the features related to Hock Law were used to design spring search algorithm (SSA). In the second chapter, spring force law has been presented. Then, in the third chapter, the general design of algorithm has been presented. In the fourth chapter, its features have been explained. In the fifth chapter, the exploration and efficiency of the proposed algorithm have been explained. Finally, the sixth chapter has concluded the study.

II. SPRING FORCE LAW

If the force that moves the object does no perform a specific action in a closed direction, that force will be conservative. Another way to recognize the conservative forces is that the action performed by force in different directions should be equal with the same start and ending points. Elastic restoring force (spring force) is an example of conservative forces [13].

Equation (1) is called spring force (Hooke's Law). Most of real springs follow Hooke's Law, if they are not strained too much [13 and 14].

$$F_s = -kx \qquad (1)$$

Here, k is constant spring force, and x is the strain or compression of spring and Fs is the spring force. Simulating Hooke's Law in an environment with discrete time, this paper designed new optimizer known as spring optimizer that will be discussed in the next section.

III. SPRING SEARCH ALGORITHM

In this paper, optimization is performed by using spring force laws in an artificial system with discrete time. System environment is the same defined range of problem. Spring force law is used as tool for information exchange. Optimizer designed to solve optimization problem, in which each problem solution can be defined as a situation in the space and its similarity to other solutions of the problem can be stated in the form of spring stiffness comparison, was used. Spring stiffness is determined according to objective function. SSA algorithm is generally explained in two steps: (1) forming an artificial system with discrete time problem environment, the initial positioning of objects, determining laws, and adjusting the parameters, and (2) passage of time up to stop time.

978-1-5386-2640-5/17/$31.00 ©2017 IEEE

A. Forming system, determining the laws, and adjustment of the parameter

In the first step, system space is defined. Environment includes a multi-dimensional coordinates in the space of the problem definition. Each point of space is a problem solution. Searcher factors are a set of objects that are connected to each other by springs. In fact, any object is connected to all other objects by a spring and each of objects has a position characteristics and stiffness coefficient of springs connected to it. Position of the object is a point in space that is the solution of the problem. The spring stiffness coefficient is determined according to the fitness of both objects connected to each other. After forming system, laws governing it are identified. We assume that only the spring force law and laws of motion are governing. Imagine the system as a set of m object. The position of each object is a point in space that is a solution to the problem. In the second equation, the position of dimension d of object i is shown by x_i^d

$$X_i = (x_i^1, \dots, x_i^d, \dots, x_i^n) \quad (2)$$

At first, the initial position of objects is defined randomly in problem space. These objects proceed to system equilibrium (solution) according to forces applied by the spring to each other. In order to calculate the spring stiffness coefficient, (3) is used.

$$K_{i,j} = K_{max}\left|F_n^i - F_n^j\right| \max(F_n^i, F_n^j) \quad (3)$$

In the equation above, $K_{i,j}$ is spring stiffness coefficient between object i and object j, Kmax is the maximum spring stiffness coefficient determined based on type of problem, and Fn is the objective function, and F_n^i, F_n^j is normalized objective functions of objects i and j. To normalize the objective function, (4) and (5) were used:

$$F'^i_n = \frac{f_{obj}^i}{\min(f_{obj})} \quad (4)$$

$$F_n^i = \min(F'^i_n) \times \frac{1}{F'^i_n} \quad (5)$$

In the equation above, f_{obj} is objective function and f_{obj}^i is objective function for object i. In a m-variable problem, it can be assumed that problem is m-dimensional and one axis is defined for each dimension, so the equivalent of each system variable can be visualized on the related axis. On each axis, the right and left strong points of object are determined according to comparison of objective function values. Strong points of each object are in fact the objects that are in more optimized position than that object. Therefore, on each axis, two resultant forces are applied to object: the resultant forces of right side and resultant forces of left side used them to calculate:

$$F_{total_R}^{j,d} = \sum_{i=1}^{n_R^d} K_{i,j}\, x_{i,j}^d \quad (6)$$

$$F_{total_L}^{j,d} = \sum_{l=1}^{n_L^d} K_{l,j}\, x_{l,j}^d \quad (7)$$

In equations above (6 and 7), $F_{total_R}^{j,d}$ is the resultant forces applied to object j from the right side and $F_{total_L}^{j,d}$ is resultant forced applied to object from left side at the dimension d. n_R^d and n_L^d are respectively strong points of right and left side of d^{th} dimension and $K_{i,j}$ and $K_{l,j}$ are spring stiffness coefficient connected between object j and strong points.

Now, by using Hooke's Law at dimension d, we have:

$$dX_R^{j,d} = \frac{F_{total_R}^{j,d}}{K_{equal_R}^j} \quad (8)$$

$$dX_L^{j,d} = \frac{F_{total_L}^{j,d}}{K_{equal_L}^j} \quad (9)$$

Here, $dX_R^{j,d}$ and $dX_L^{j,d}$ are respectively replacement value of object j to right and left side at the dimension d. Therefore, we will have:

$$dX^{j,d} = dX_R^{j,d} + dX_L^{j,d} \quad (10)$$

$dX^{j,d}$ is the final displacement value of object j in line with dimension d, which its value can be positive and negative according to (10). Now, we have:

$$X^{j,d} = X_0^{j,d} + r_1 \times dX^{j,d} \quad (11)$$

In equation above, $X^{j,d}$ relates to new equilibrium point and location of dimension d of system and object j. In addition, $X_0^{j,d}$ is the initial location of object i equilibrium in line with dimension d. Here, r1 is a random number with uniform distribution in the range [0-1], used to maintain the random state of the search.

B. Passage of time and updating the parameters

At the beginning of system forming, any object is randomly placed at a point in space that is solution of the problem. At any point in time, objects are evaluated. Then, displacement of each object is obtained after a calculating the 3 to 10 equations. System parameter is spring stiffness coefficient, which is updated at each step according to equation (3). The stop condition can be determined after a certain time. Different steps of spring force algorithm are as follows:

1. Determining the system environment and initializing.
2. The initial placement of objects
3. Evaluation and Normalization of objects fitness
4. Updating the parameter K
5. Forming the spring force laws for each of objects
6. Calculating the displacement of objects
7. Updating the displacement of objects
8. Unless the stop condition has been met, repeat the steps 3 to 7
9. End

IV. CHARACTERISTICS OF THE PROPOSED ALGORITHM

In algorithm above, it has been tried that a method to be created for optimization by using spring force law. In this algorithm, a set of objects searches the space randomly. Spring force was used as tool for information exchange. Each object achieves to approximate understanding of surrounding space affected by position of other objects. The algorithm should be directed in such a way that the position of objects to recover over time. Used strategy in this regard is adjustment of the spring stiffness factor. Therefore, a spring with higher stiffness coefficient is connected to objects that have better fitness function and drag other objects toward itself. As a result, for any object, force proportional to size of that object is applied. In addition, objects that are at better positions should have shorter and slower steps. To achieve to this goal, we attribute spring with larger stiffness coefficient to better objects. This makes that any object that has better fitness search the space around it more accurately. This issue is like adjustment of learning rate in neural networks, which is adaptive for each object here. Stiffness coefficient of springs and as result force of springs make smaller over time. As we know that objects have been accumulated around the better positions over time, and it is needed that space with smaller steps and with more accuracy to be searched, we make the spring stiffness smaller over time. Figure. 1 is a display of forces applied to system and algorithm performance.

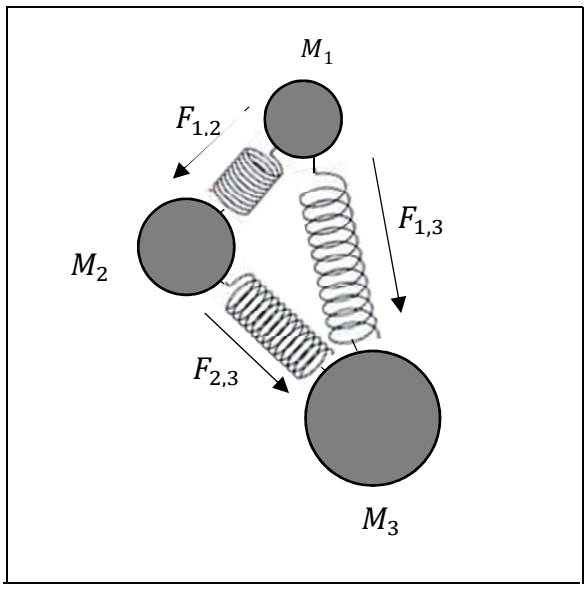

Figure1. Each object is displaced according to spring forces applied to it in the SSA algorithm.

V. EXPLORATION AND EFFICIENCY OF SSA ALGORITHM

One of the recommendations to improve the algorithm is improving detection power of algorithm. Two issues are involved in optimization, including exploration and efficiency. In the exploration issue, any optimization algorithm should have the power of problem space exploration at good level, and it should not be limited only to some areas. In efficiency issue, the ability of the algorithm in detection of optimal areas is discussed. Need for comprehensive search of space is felt and algorithm should emphasize on better search of space in initial iterations, but the ability to detect the algorithm becomes more evident over time and algorithm should position by using the findings of swarm in the optimal points [12 and 15]. The mentioned algorithm has the proper search power of the space with considering the number of objects at appropriate level. To improve and speed up the detection power algorithm, the spring force is effective. To achieve this goal, spring stiffness coefficient and thus the spring force are applied between objects. Therefore, the value of spring stiffness coefficient is controlled as algorithm proceeds. Proper value is selected for it at the beginning, while its value decreases as time passes, so that it reaches to minimum level at the end. The equation considered for spring stiffness coefficient that is Equation 3 has this feature.

VI. SIMULATION AND RESULTS

To evaluate the proposed method, this algorithm, genetic algorithm and swarm algorithm have been implemented in solving the minimum finding problems in Table 1 in equal conditions. For the state n =30, number of swarm is 50, and the results for 1000 iterations of algorithm are presented in Table 2. For comparison, the average merit and the best solution observed so far are calculated. These parameters were calculated for 20 times of independent implementation of the program, and the median of the results was obtained. The results indicate better efficiency of spring force algorithm. In simulation of PSO, to update the particle velocity, Equation 12 was used. In this equation, c1= c2=2 and w decreases linearly from 9.0 to 2.0. In this equation, $V_i^d(t)$ is the velocity of dimension d of particle t and r1 and r2 are random numbers with uniform distribution between zero and one. In addition, gbest is the best position found by community so far, and $pbest_i$is the best position that i^{th} particle has achieved to it so far.

$$V_i^d(t+1) = w(t)V_i^d(t) + c_1 r_{1i}(t)\left[pbest_i^d(t) - X_i^d(t)\right] + c_2 r_{2i}(t)[gbest^d(t) - X_i^d(t)] \quad (12)$$

In order to optimize objective functions introduced, as can be seen in Table 2, the proposed SSA algorithm showed better performance and acceptable results in comparison with RGA and PSO algorithms. In order to evaluate the progress of the optimization process, the way to achieve the optimal solution has been shown for all functions in Figures 2 to 6. Results indicate the higher convergence of the SSA algorithm, compared to PSO and RGA algorithms.

Table 1. Minimizing functions used in tests, n specifies the dimension of the problem. Optimal value of all functions is zero.

Test functions	R^n
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
$f_3(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	$[-100, 100]^n$
$f_4(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
$f_5 = \sum_{i=1}^{n} i x_i^4 + random[0,1]$	$[-1.28, 1.28]^n$

Table 2. Results of implementing of the optimization on functions of Table 1

Functions	Criteria	RGA	PSO	SSA
f_1	Best solution mean	23.13	1.9×10^{-3}	1.5×10^{-11}
	Fitness mean	23.45	5.1×10^{-2}	6.5×10^{-9}
f_2	Best solution mean	1.08	2.1	9.4×10^{-8}
	Fitness mean	1.08	2.1	2.6×10^{-7}
f_3	Best solution mean	24.02	1.1×10^{-3}	4.6×10^{-12}
	Fitness mean	24.53	6.7×10^{-3}	3.1×10^{-11}
f_4	Best solution mean	5.7×10^{3}	4.2×10^{3}	4×10^{-3}
	Fitness mean	5.7×10^{3}	2.8×10^{3}	8.2×10^{-2}
f_5	Best solution mean	0.07	0.04	0.019
	Fitness mean	0.57	1.04	0.612

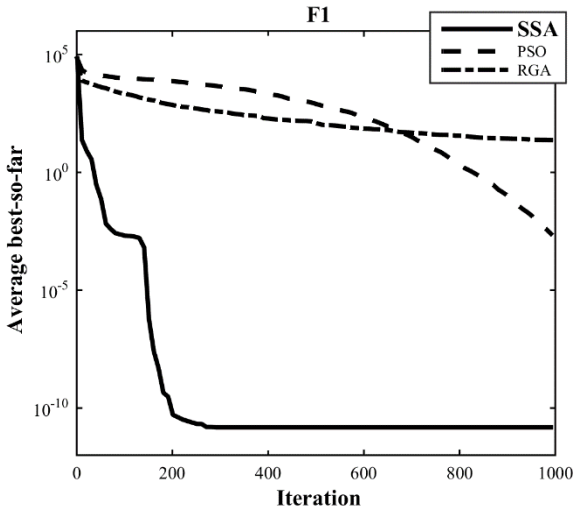

Figure 2. Performance of SSA, GA and PSO for f_1

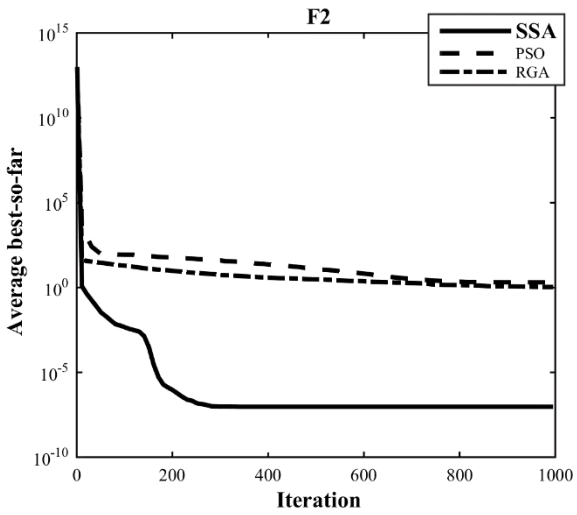

Figure 3. Performance of SSA, GA and PSO for f_2

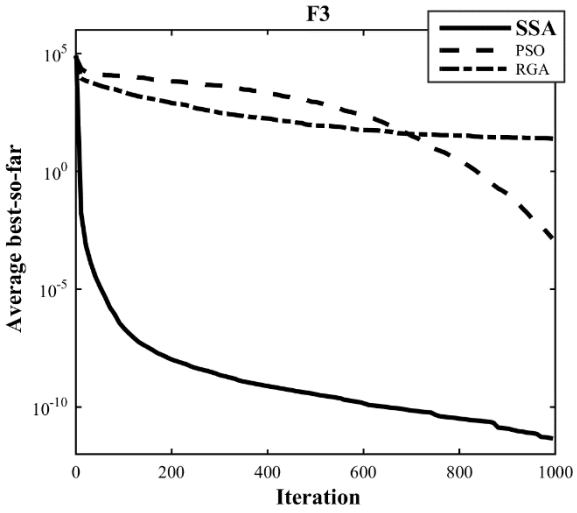

Figure 4. Performance of SSA, GA and PSO for f_3

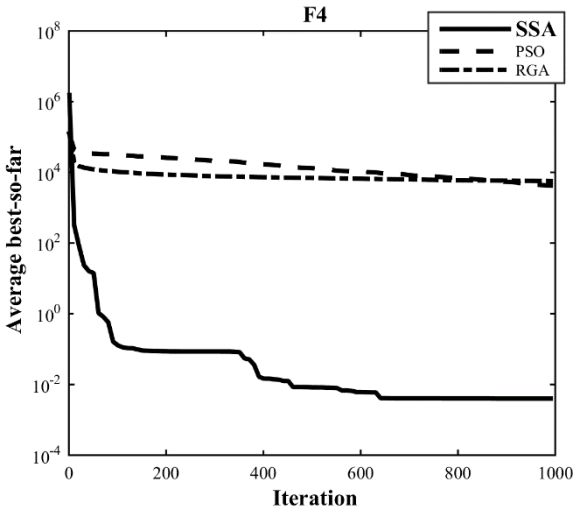

Figure 5. Performance of SSA, GA and PSO for f_4

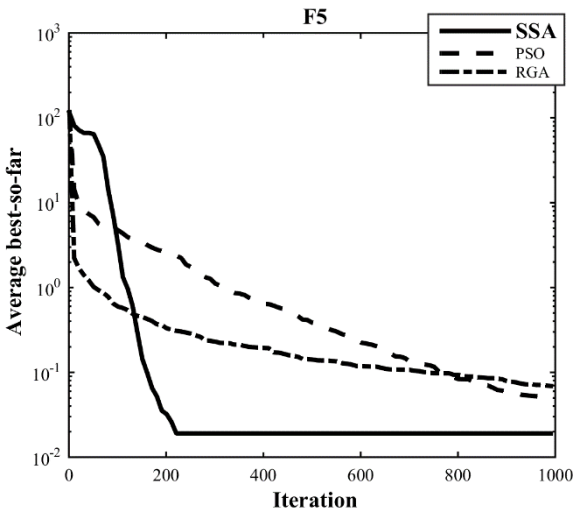

Figure 6. Performance of SSA, GA and PSO for f_5

VII. CONCLUSION

Nowadays, accident-based algorithms are used widely for optimization. Most of these algorithms were inspired by physical processes or behavior of creatures. In this paper, a new approach of using spring force law was examined. The proposed algorithm was inspired by the laws of nature and the results show the appropriate performance in optimization. Results of comparison of the proposed method with PSO algorithm and genetic algorithm show the better performance of spring algorithm.

REFERENCES

[1] K.S .Tang, K.F.Man, S.Kwong and Q.He, " Genetic algorithms and their applications ", IEEE Signal Processing Magazine 13 (6), 1996.

[2] S.Kirkpatrick, , C.D.Gelatto and M.P.Vecchi, "Optimization by simulated annealing", Science 220 (4598), 671–680. 1983.

[3] J.D. Farmer, N.H.Packard and , A.S.Perelson," The immune system, adaptation, and machine learning", Physica D 22, 187–204. 1986.

[4] M.Dorigo, V.Maniezzo, and A.Colorni, "The Ant System: optimization by a colony of cooperating agents.", IEEE Transaction on systems, Man, and Cybernetics-part B, vol. 26, no.1, 1996, pp. 1-13.

[5] J.Kennedy and R.C.Eberhart, "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995.

[6] FazelZarandi, M.H., Hemmati, A., Davari, S. "The multi-depot capacitated location-routing problem with fuzzy travel times." Expert Systems with Applications, vol. 38, no. 8, pp. 10075–10084 (2011).

[7] Mitra, S., Banka, H. "Multi-objective evolutionary biclustering of gene expression data." Pattern Recognition, vol. 39, no. 12, pp. 2464–2477(2006).

[8] Zahiri, S.H. "Swarm Intelligence and Fuzzy Systems." Nova Science Publishers, USA (2010).

[9] Darby, S., Mortimer-Jones, T.V., Johnston, R.L.,Roberts, C, "Theoretical study of CuAunanoalloy clusters using a genetic algorithm." Journal of Chemical Physics, vol. 116, no. 4, pp. 1536–1550 (2002).

[10] CoelloCoello, C.A., Luna, E.H., Aguirre, A.H. "Use of Particle Swarm Optimization to Design Combinational Logic Circuits. In: Evolvable Systems." From Biology to Hardware, Springer Berlin, Heidelberg, pp. 398–409 (2003).

[11] Wolpert, D.H.,Macready, W.G. "No free lunch theorems for optimization." Evolutionary Computation, IEEE Transactions on, vol. 1, no. 1, pp. 67–82 (1997).

[12] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S. "GSA: A Gravitational Search Algorithm." Information Sciences, vol. 179, no. 13, pp. 2232–2248 (2009).

[13] D.Holliday, R.resnick and J.Walker, Fundamentals of physics, John wiley and sons, 1993.

[14] Lee, C.K.; Tan, S.C.; Wu, F.F.; Hui, S.Y.R.; Chaudhuri, B, " Use of Hooke's Law for Stabilizing Future Smart Grid – The Electric Spring Concept" Energy Conversion Congress and Exposition (ECCE), 2013 IEEE.

[15] Eiben, A.E., Schippers, C.A, "On Evolutionary Exploration and Exploitation." FundamentaInformaticae, vol. 35, no. 1-4, IOS Press, Amsterdam (1998).