Periodicals of Engineering and Natural Sciences ISSN 2303-4521
Vol.5, No.3, November 2017, pp. 378~386
Available online at: http://pen.ius.edu.ba

Solving Constrained Optimization Problems with Sine-Cosine Algorithm

***Simge Ekiz*[1], *Pakize Erdoğmuş*[1], *Büşra Özgür*[1]**
[1]*University, Department of Computer Engineering, Duzce, Turkey*

Article Info

Article history:

Received Jun 1st, 2017
Revised Aug 20th, 2017
Accepted Oct 18th, 2017

Keyword:

Sine-Cosine Optimization Algorithm,
Constrained Optimization Problems,
Penalty Method

ABSTRACT

Optimization algorithms aim to find the optimum values that give the maximum or minimum result of a function under given circumstance.

There are many approaches to solve optimization problems. Stochastic population-based optimization approaches tend to give the best results in a reasonable time. Two of the state-of-art stochastic optimization algorithms are Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). In addition, Sine-Cosine Algorithm is one of the recently developed stochastic population-based optimization algorithms. It is claimed that Sine-Cosine has a higher speed than the counterparts of it. Moreover, Sine-Cosine Algorithm occasionally outperforms other optimization algorithms including GA and PSO. This algorithm is successful because it can balance exploration and exploitation smoothly.

In the previous studies, the above-mentioned algorithms were evaluated and compared to each other for the unconstrained optimization test functions. But there is no study on constrained optimization test problems. In this study, we aim to show the performance of Sine-Cosine Algorithm on constrained optimization problems. In order to achieve this, we are going to compare the performances by using well-known constrained test functions.

Corresponding Author:

Simge Ekiz,
Duzce University, Department of Computer Engineering, Duzce, Turkey.
Email: simgeekiz@duzce.edu.tr

1. Introduction

Optimization can be defined as finding the most effective and highest achievable performance under the given limitations. Mathematically speaking, Optimization is finding the minimum or maximum of a function subject to the constraints. A set of values that satisfies all the constraints of an optimization problem creates a feasible solution. The optimization technique tries to find the optimum solution from all of these feasible solutions [1].

Optimization problems can be found in every area of life because all living things tend to do the best. For example, Birds fly in ‘V’ shape to reduce the energy consumption, another example fish moves in flocks to benefit from defense against predators [2]. Optimization problems have been a topic since 1960’s. In these years optimization problems have been tried to solve by classical mathematical methods. Deterministic methods have a great advantage that they find global optima. Unfortunately, they cannot solve all nonlinear problems. With classical optimization, only limited problems can be solved. The inadequacy of classical methods has forced scientists to search for new methods. To find the optimum, solution stochastic algorithms are developed. These algorithms sample the search space without exploring it thoroughly. Stochastic computation techniques have received a great deal of attention regarding their potential as optimization techniques for complex problems. As a result, the development of stochastic algorithms has begun.

It is impossible to develop one way to solve all the nonlinear problems. There are a lot of methods proposed. Stochastic algorithms can find promising solutions for difficult optimization problems, but there is no guarantee that optimal solutions can be reached all the time. Stochastic algorithms are good at solving most of the real world problems which are nonlinear and multimodal[1].

The general nonlinear programming problem is defined as follows[3];

Minimize $f(x), \quad x = (x_1, \ldots, x_n) \in R^n$

Subject to the constraints $g_i(X) \geq 0$

$h_i(X) = 0$

Where $g_i(X)$ and $h_i(X)$ are constraints that are required to be satisfied.

The optimization problem is based on finding the optimum value of the objective function, and if there are no constraints on the variables, these problems are called unconstrained optimization problems. The solution of unconstrained optimization problems is easier than the constrained ones. However, most real-life problems are constrained. It is necessary to find the best value of a constraint optimization problem such as resource constraint, time constraint, cost constraint, design constraint according to these conditions.

1.1. Related Works

In 1994 Joines and Houck solve four test cases of constrained optimization problems with the genetic algorithm. They transform constrained optimization problems to the unconstraint optimization by using the penalty method. They aim at reaching the feasible solution of genetic algorithm by giving appropriate value according to the number of generations[4]. In 1996 Michalewicz and Schoenauer present several constraint-handling techniques for optimization problems. The first one based on feasibility of solutions, the second one is penalty based, the third method makes a clear distinction between feasible and infeasible and fourthly hybrid methods. Moreover, they provide 11 test cases which we use in this study to make our experiments[3]. In 2002 Hu and Eberhart applied one of the constraint handling methods which based on preserving feasibility of solutions. They test particle swarm optimization algorithm on the same test cases[5]. In 2005 Yeniay go over all constraint handling penalty based techniques for the genetic algorithm. He mentions their advantages and disadvantages. He emphasizes the importance of setting appropriate values of the penalty parameters[6].

1.2. Dealing with constraints

All the stochastic algorithms are directly suited to unconstrained optimization problems. Applying these algorithms to constrained optimization problems has always been a problem. In real life problems such as engineering design problems are constrained optimization problems and constraints has a great effect on the optimization performance[7]. Fortunately, many constrained optimization algorithms can be transformed to the unconstrained case, often with the use of a penalty method.

Penalty function method is common because of its simple principle and easy implementation. We modify the objective function in such a way that it penalizes any violation of the constraints.

Penalty function method can be formulated as follows[8];

Minimize $f(x)$

Subject to the constraints $g_i(X) \geq 0 \qquad i = 1,2,3, \ldots\ldots m$

The equivalent unconstrained optimization problem can be stated as;

Minimize $F(x) = f(x) + P(x)$

Maximize $f(x)$

Subject to the constraints $g_i(X) \geq 0 \quad i = 1,2,3, \ldots\ldots m$

The equivalent unconstrained optimization problem can be stated as;

Maximize $F(x) = f(x) - P(x)$

In minimization problems, we include the penalty function which adds a high cost to the objective function. In maximization problems, we subtract the penalty function from the objective function.

In this study, we implement two of the state-of-art algorithms that we mentioned to the well-known constraint test cases. In addition to these algorithms, we also implement Sine Cosine Algorithm that developed in 2016[9]. The above-mentioned algorithms were directly suited to unconstrained optimization test functions. To deal with constraints we use the penalty function method due to its popularity and easy implementation [7]. Penalty method adds the penalty term to the objective function for any violation of the constraints [8], [10]. We aim to show the performance of Sine-Cosine Algorithm on constrained optimization problem and compare results with genetic algorithm and particle swarm optimization algorithm.

2. Background theories

2.1. Genetic Algorithm (GA)

```
Pseudo code for Genetic Algorithm;
Begin
    t ← 0;
    InitializePopulation[P(t)];
    EvaluatePopulation[P(t)];
    while not termination do
        P'(t) ← Selection[P(t)];
        P(t+1) ← ApplyGeneticOperators[P'(t)uQ];
        EvaluatePopulation[P(t+1)];
        t ← t+1;
    end while
    return BestSolution
end
```

Genetic algorithm mimics the biological evaluation. It starts with creating a population randomly. And population can be described as a group of individual solutions. In each iteration, the algorithm chooses some solutions from the current population as parents according to their fitness values to form a new generation. The new generation is created by applying genetic operators such as crossover and mutation. Next, these new generations are evaluated, and this process is going on until the termination condition is met. And the population evolves toward an optimal solution.

2.2. Particle Swarm Optimization Algorithm (PSO)

```
Pseudo code for Particle Swarm Optimization Algorithm;
Begin
    t ⟵ 0;
    InitializeParticles[P(t)];
    EvaluateParticles [P(t)];
    while not termination do
        t ⟵ t+1;
        Select pBest for each Particle;
        Select gBest from P(t-1);
        CalculateParticleVelocity[P(t)];
        UpdateParticlePosition[P(t)];
        EvaluateParticles [P(t)];
    end while
    return BestSolution
end
```

Particle Swarm Optimization is another population optimization technique developed by Eberhart and Kennedy in 1995. It starts with initializing the population of random solutions called particles. In PSO the particles have velocity values alongside the fitness values. In every iteration, gbest and pbest are selected. Pbest is the best solution has achieved so far by a particle. Gbest is the best value obtained so far by any particle in the population. Next, the velocities of the particles are calculated using pbest and gbest values. Then particle positions are updated based on the velocities. Thus, the particles follows the best particle in the search space.[11]

2.3. Sine Cosine Algorithm (SCA)

```
Pseudo code for Sine Cosine Algorithm;
begin
    InitializeSearchAgents[X];
    while t<maximum number of iterations do
        EvaluateSearchAgents[X];
        UpdateBestSolution(P=X*);
        Update $r_1 r_2 r_3 r_4$ ;
        UpdateSearchAgentPosition;
    end while
return BestSolution
end
```

Sine cosine algorithm can be described as the following formula;

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * \sin(r_2) * \left|r_3 P_i^t - X_i^t\right|, & r_4 < 0.5 \\ X_i^t + r_1 * \cos(r_2) * \left|r_3 P_i^t - X_i^t\right|, & r_4 \geq 0.5 \end{cases}$$

As another population-based optimization algorithm sine cosine algorithm also starts with initializing random solutions called search agents. Sine cosine algorithm uses 4 variables to tune. These are r variables. r_1 decides that search agent is going to do whether exploration or exploitation. All stochastic algorithms are both exploration and exploitation but it is important to balance these. r_2 decides how far the solution's movement should be. r_3 assign a random weight and r_4 decided whether sine or cosine formula is going to be used.

In every iteration, the solutions are evaluated by using the fitness function and the algorithm assigns the best solution obtained so far as the destination point. Next, the r variables are updated.

Search agent positions are updated based on the r variables and the best solution. Thus the potential solutions follow the best solution in the search space [9].

3. Experiments

We use global optimization toolbox in Matlab© which includes both genetic algorithm and particle swarm optimization algorithms with default values. And we implement Sine cosine algorithm with 30 search agents and 1000 iterations. The algorithms are run 30 times on the popular constraint optimization test problems that Michalewicz and Schoenauer first presented. We get these test problems from Kyoto University global optimization test problems web site[13]. We implement penalty method to these test cases.

The test problems are given below;

G1	Min $f(x) = \sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - 5\sum_{i=5}^{13} x_i$
G2	Max $f(x) = \left\lvert \frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^4(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right\rvert$
G4	Min $f(x) = 5.3578547x_3^2 + 0.8356892x_1x_5 + 37.293239x_1 - 40792.141$
G5	Min $f(x) = 3x_1 + 10^{-6}x_1^3 + 2x_2 + \frac{2}{3} * 10^{-6}x_2^3$
G6	Min $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$
G7	Min $f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2$ $+2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$
G8	Max $f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$
G9	Min $f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4$ $-4x_6x_7 - 10x_6 - 8x_7$
G10	Min $f(x) = x_1 + x_2 + x_3$
G11	Min $f(x) = x_1^2 + (x_2 - 1)^2$
G13	Min $f(x) = e^{x_1x_2x_3x_4x_5}$

4. Experimental Results

Results are shown in figures. The red lines represent the best-known values gathered in Kyoto University Global Optimization Website.

Sine cosine algorithm, genetic algorithm, and particle swarm optimization algorithm are represented by respectively by orange, pink and blue bars. In terms of accuracy mostly genetic algorithm gives the best results. SCA algorithm is outperformed by the others or results as the same.

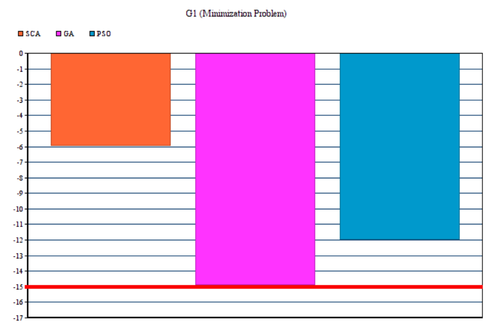

Figure 1. G1 Test Function(Minimization)

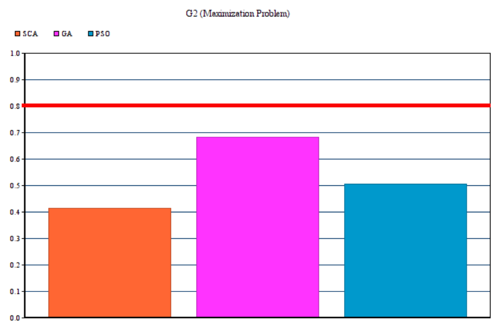

Figure 2. G2 Test Function(Maximization)

In G1 global optimization test case, genetic algorithm outperforms the other two algorithms. G1 is minimization problem and GA reaches the minimum value which is -15.

In G2 is maximization problem and in these experiments, any of the algorithms reach the maximum value gathered in Kyoto University Global Optimization Website so far.

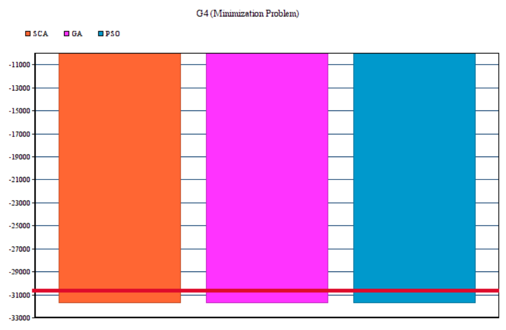

Figure 3. G4 Test Function(Minimization)

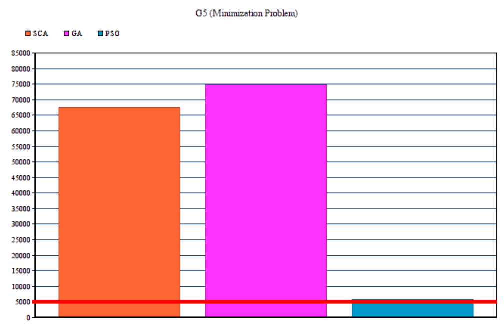

Figure 4. G5 Test Function(Minimization)

All the algorithms find the optimum value in G4 and G11 test cases. Like in these situations it is better to choose the algorithm which takes a shorter time. In G5 only PSO find the minimum optimal value. In G6, SCA and GA are both find the minimum value which updates the minimum value found up to now. Moreover, SCA is not good at solving problems like G7 but in the G8 test case, SCA outperforms both GA and PSO. In G9, G10, G13 test cases SCA is outperformed by the others.

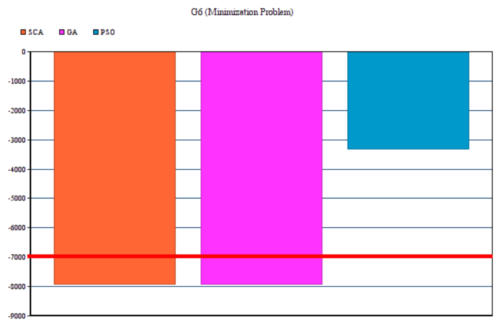

Figure 5. G6 Test Function(Minimization)

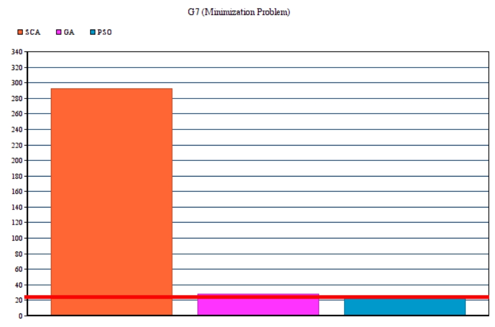

Figure 6. G7 Test Function(Minimization)

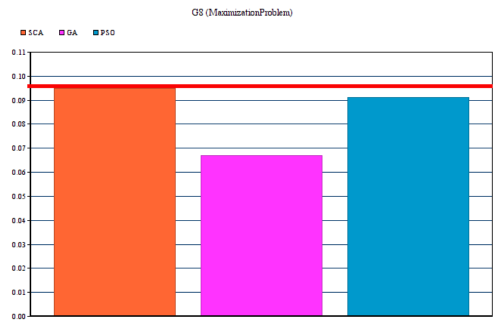

Figure 7. G8 Test Function(Maximization)

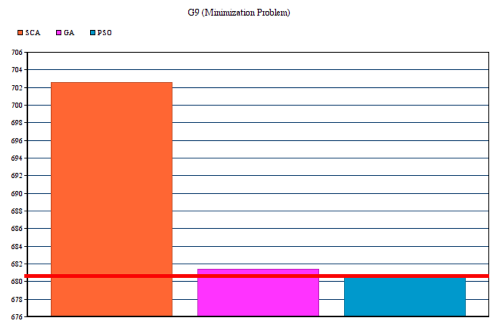

Figure 8. G9 Test Function(Minimization)

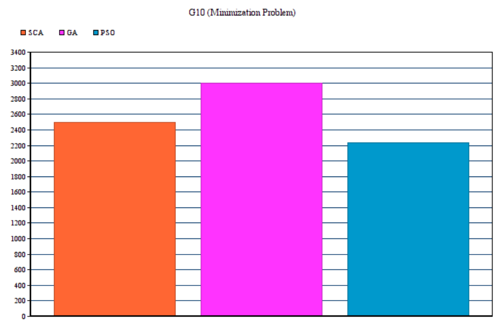

Figure 9. G10 Test Function(Minimization)

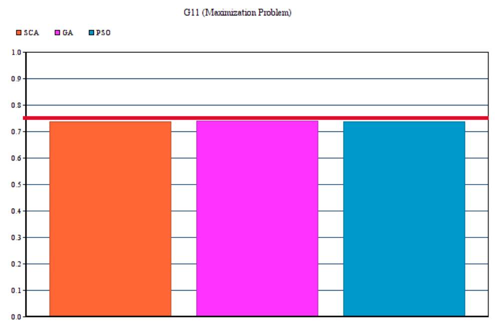

Figure 10. G11 Test Function(Maximization)

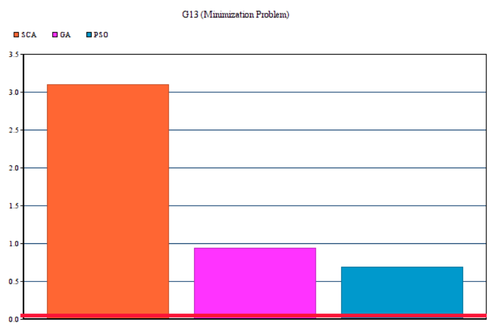

Figure 11. G13 Test Function(Minimization)

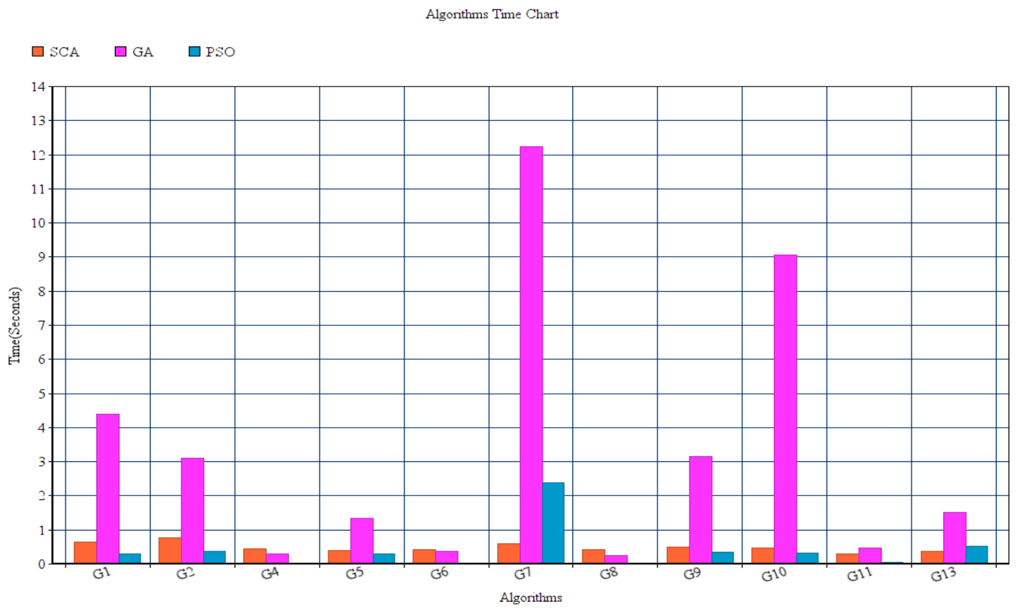

Figure 12. The time taken by the algorithms for each test case

This graph demonstrates the time taken by the algorithm for each problem. In most of the cases, the slowest algorithm is a genetic algorithm. Except two of the cases, PSO is the fastest one. And SCA takes slightly more time than the PSO. Mostly both PSO and SCA finish under a second.

5. Conclusions and Discussion

In this study, we optimized well known constrained optimization problems with using the recently developed sine cosine algorithm. We deal with constraints with the help of penalty method. Then we compared SCA algorithm results with GA and PSO. In terms of accuracy GA gives the best results but in terms of speed PSO and SCA are faster. Although SCA gives better result in unconstrained optimization problems, it does not perform well on constrained optimization problems. But SCA is a new algorithm and it might give better results by making further improvement in the algorithm. Also, it might give better results when we change the method of handling constraints.

REFERENCES

[1] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, 2016.

[2] "Swarm Behaviour." [Online]. Available: https://en.wikipedia.org/wiki/Swarm_behaviour.

[3] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, 1996.

[4] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1994, pp. 579–584.

[5] X. Hu and R. Eberhart, "Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization," *Optimization*, vol. 2, no. 1, pp. 1677–1681, 2002.

[6] O. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Math. Comput. Appl.*, vol. 10, no. 1, pp. 45–56, 2005.

[7] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Eng. Appl. Artif. Intell.*, vol. 20, no. 1, pp. 89–99, 2007.

[8] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained Optimization Via Genetic Algorithms," *Simulation*, vol. 62, no. 4, pp. 242–253, 1994.

[9] S. Mirjalili, "SCA: A Sine Cosine Algorithm for solving optimization problems," *Knowledge-Based Syst.*, vol. 96, pp. 120–133, 2016.

[10] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," *Bull. Am. Math. Soc.*, vol. 49, no. 1, pp. 1–24, 1943.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.

[12] J. Brownlee, "Particle Swarm Optimization." [Online]. Available: http://www.cleveralgorithms.com/nature-inspired/swarm/pso.html. [Accessed: 26-Apr-2017].

[13] A. R. Hedar, "Global Optimization Test Problems." [Online]. Available: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.