Optimizing Gradient Methods for IoT Applications

Hosseini, Eghbal; Reinhardt, Line; Rawat, Danda B.

Published in:
IEEE Internet of Things Journal

DOI:
10.1109/JIOT.2022.3142200

Publication date:
2022

Document Version
Peer reviewed version

Citation for published version (APA):
Hosseini, E., Reinhardt, L., & Rawat, D. B. (2022). Optimizing Gradient Methods for IoT Applications. *IEEE Internet of Things Journal*, *9*(15), 13694-13704. https://doi.org/10.1109/JIOT.2022.3142200

Optimizing Gradient Methods for IoT Applications

Eghbal Hosseini, Line Reinhardt, Danda B. Rawat *Senior Member, IEEE*

***Abstract*—Solving Linear Programming (LP) and Non-Linear Programming (NLP) problems are momentous because of their wide applications in real-life problems. There is no unified way to find the global optimum for NLPs. But on the other hand, Simplex Algorithm, as the dominating methodology for LPs for several decades moves only on the boundary (vertices) and ignores the vast majority of the feasible region in the process of searching. In this paper, we study two gradient-based methodologies that explore the whole feasible region, which guarantee faster convergence rates for both LP and NLP optimization problems including IoT problems such as Software Defined Internet of Vehicles (SDIoV) and Vehicular Ad hoc Networks (VANETs). The Gradient-Simplex Algorithm (GSA) for LPs, which moves inside the feasible region in the gradient direction at first to reduce the search space and then explores the reduced boundary to find an optimal solution. The Evolutionary-Gradient Algorithm (EGA), on the other hand, is for NLPs and uses an evolutionary population to estimate gradients by evolving to find better solutions in steps. Based on extensive simulations, obtained numerical results show that both approaches provide efficient solutions and outperform the state-of-the-art methods on optimization problems with large feasible spaces. Comparative results of applying the GSA on SDIoV and VANETs with different sizes are included.**

***Index Terms*—Gradient Function, Simplex Algorithm, Evolutionary Population, Feasible Region, Internet of Things.**

I. Introduction

The Simplex Algorithm has, for many years been, the most widely known methodology for solving Linear Programming (LP) problems. Recent approaches for solving LPs inherit the structural properties of the Simplex Algorithm. Such algorithms include but are not limited to upper bounding the number of basic solutions of the Simplex Algorithm [1], algorithms based on a hybrid of meta-heuristic and modified Simplex Algorithm [2], [3], other algorithms can be found in [4]–[7]. Nevertheless, the Simplex Algorithm searches for the optimal solution by exploring the vertices of the polyhedron, therefore the efficiency of the algorithm will be lost by moving along the boundary of the feasible region.

Nonlinear programming (NLP), on the other hand, has many applications in practice, such as in the area of regression [8]–[10], portfolio [11], energy [12], etc. The gradient is a key factor to consider in NLP; most NLP methodologies rely on a gradient or an estimate of a gradient to define the search direction, e.g., Steepest Descent/Ascent Method, Newton's Method, etc. Both LPs and NLPs often occur in IoT applications among others the problems Software-Defined Internet of Vehicles (SDIoV) [13] and Vehicular Ad hoc Networks (VANETs) [14].

Recently researchers have with success applied heuristic approaches for solving specific cases such as the vehicle connectivity problem [13]. Additionally, researchers have solved optimization problems using simulation of several algorithms based on the behavior of animals and insects, natural phenomena, or scientific theories [15]–[25]. Some of these proposed algorithms are: particle swarm optimization (PSO) [15], artificial bee colony algorithm [16], krill herd algorithm [17], laying chicken algorithm (LCA) [13], volcano eruption algorithm (VEA) [18], multiverse algorithm (MVA) [19], Covid-19 optimizer algorithm (CVA) [20] and grey wolf optimizer [21], bat algorithm [22], social spider optimization [23], chicken swarm optimization (CSO) [24], Fire Fly (FF) [25]. Recently, an algorithm based on a hidden Markov model and an ant colony optimization (ACO) has been proposed to answer the service composition issue by enhancing the QoS [26], an agent-based algorithm for achieving a distributed resources organization in an IoT environment [27] and a secure outsourcing algorithm of bi-linear pairings, which does not require pre-computations has been proposed [28].

This paper proposes new methodologies for solving LPs and NLPs with a focus on IoT problems with different data formats and different network sizes. The approaches apply the ideas of Gradient-based methods to generate moving directions inside the feasible region. The Gradient Simplex Algorithm (GSA) method for LPs avoids the boundary search of the Simplex Algorithm but instead, searches in a continuous space within the feasible region and based on computational results, see Instances 1,2 and Table I, and results for IoT problems in Table IV, Table V, GSA has better results than other algorithms including the Simplex Algorithm. When reaching the boundary of the feasible region, GSA then, applies the ideas of the Simplex Algorithm for locating the optimal vertex. The Evolutionary-Gradient Algorithm (EGA), on the other hand, is for NLPs. EGA uses an evolution population to explore the local vertices and to evolve to better solutions based on the estimate gradients. According to the numerical tests on representative examples, the proposed algorithms converge faster than existing approaches on both LPs and NLPs for problems with many constraints or boundary points, especially on IoT problems with considering all of connectivity, reliability, packet delivery ratio, and delay variance of the route. Because GSA solves LPs, it is efficient for some IoT problems like SDIoV. Also, EGA is an efficient algorithm to solve all nonlinear and nonconvex linear programming, so it can solve VANETs as NLPs. The results have been shown in tables IV

Eghbal Hosseini and Line Reinhardt are with Department of People and Technology Roskilde University, Denmark (e-mail:kseghbalhosseini@gmail.com, hosseini@ruc.dk, liner@ruc.dk).

Danda B. Rawat is with Department of Electrical Engineering and Computer Science, Howard University, Washington, DC, USA(e-mail:db.rawat@ieee.org).

The researchers can access the implementation and programming code in https://github.com/eghbal11/Eghbal/blob/master/GSA%20and%20EGA

and V.

Some of novelties of GSA and EGA are as follows:

1) GSA is much faster than simplex based on (17).
2) EGA is an evolutionary algorithm which unlike other evolutionary algorithms uses gradient concepts.
3) There are many heuristic algorithms in the literature to solve IoT which can solve a specific form of the problem, GSA and EGA can solve different types of IoT.

The rest of the article is organized as follows: Section II proposes the conceptual ideas of GSA as a combination of typical Gradient Method and Simplex Algorithm, which is then extended in Section III with Evolutionary populations. Computational complexity and convergence and convexity are discussed in Sections IV and V respectively. Computational Results are presented in Section VI and the conclusion is found in Section VII.

II. Gradient-Simplex Algorithm (GSA) for LPs

The fundamental idea of the Gradient-Simplex Algorithm (GSA) is to replace moving along the boundary of the Simplex algorithm by a movement across the polyhedron along the Gradient direction. We begin our discussion by presenting the Gradient-Simplex Algorithm (GSA) in this section and then extend it with evolutionary populations in the next.

Consider the following LP in canonical form:

$$\begin{aligned} &\max cx \\ &s.t Ax \leq b \\ &x \geq 0 \end{aligned} \tag{1}$$

where A is a $m \times n$ matrix, x and b are two $n \times 1$ vectors, c is a $1 \times m$ vector.

The GSA works in two phases. In the first phase, moves in the direction of the gradient of the objective from the origin. In the second phase, the GSA applies the Simplex algorithm on the feasible solutions found in the precious phase in order to find the optimal solution.

1) Phase 1. Moving in Gradient Direction: In this phase, we calculate the gradient of the objective function and find the intersection of the gradient direction with the boundary of the feasible region. By doing this the active constraint bounding the optimal solution is identified. Consider problem (1) which has m constraints and n variables. The algorithm moves in the direction of the gradient vector until it meets the first hyperplane that is defined by a constraint. Let

$$x = \gamma_1 \cdot c = \gamma_1 \cdot (c_1, c_2, ..., c_n) \tag{2}$$

where γ is a constant representing the maximum moving step in the Gradient direction without violating any constraints. Then the following constraints must hold:

$$Ax = A\gamma_1 c = \gamma_1 (\sum_{j=1}^{n} a_{ij} c_j) \leq b_i, i = 1, 2..., m \tag{3}$$

Now let

$$s_i = \sum_{j=1}^{n} a_{ij} c_j, i = 1, 2..., m \tag{4}$$

The constraint which intersects the gradient vector first can be found as the minimum intercept of the m constraints, which is:

$$r = \min(\frac{b_i}{s_i} | s_i > 0, i = 1, 2..., m) \tag{5}$$

In case when (5) is not empty and the origin is feasible, moving further in the same direction will violate at least the constraint $\hat{i}$ so we are already on the boundary of the feasible region and the constraint $\hat{i}$ should be active in the optimal solution. If the origin is not in the feasible region the method uses the M-large method to find an initial feasible solution, and move in the gradient direction from it. Algorithm 1 shows the pseudo-code for the first phase procedures.

Algorithm 1 Pseudo code of initial solution and the first population

```
Initialization of Variables:
[m, n] = size(A)
t = n
A = [Aeye(m)]
b = b(:)
c = −c(:)′
c = [czeros(1, m + 1)]
A = [Ab]
for i = 1 to m do
    d(i) = 0
    for j = 1 to n do
        d(i) = d(i) + A(i, j) ∗ c(j)
    end for
end for
min = b(1)/d(1), r = 1
for j = 1 to m do
    k(j) = b(j)/d(j)
    if k(j) < min, d(j) > 0 then
        min = k(j), r = j
    end if
end for
```

2) Phase 2. Continuing by Simplex Algorithm: Next let's define the Simplex based part of the algorithm and thus continue the solution process by applying the Simplex Algorithm in order to locate the correct vertex for the optimal solution. As discussed in phase 1, the constraint $\hat{i}$ should be active in the optimal solution, which means the corresponding slack variable should be set to zero before continuing with the Simplex Algorithm and remains zero for all the following iterations. Therefore, the slack variable should be removed from the basis.

Let $r = \frac{b_i}{s_i}$, be the length of movement in the direction of the gradient. The pseudo code of this phase is shown in Algorithm 2.

3) Steps of the algorithm: The main steps of the GSA are proposed as follows:

Algorithm 2 Pseudo code of continuing by simplex algorithm

```
max = c(1), s = 1
for i = 1 to n + m do
    if c(i) > max, A(r, i) > 0 then
        max = c(i), s = i
    end if
end for
A = vertcat(c, A)
row = r + 1, col = s
A(row, :) = A(row, :)/A(row, col)
for j = 1 to m + 1 do
    if i = row then
        A(i, :) = A(i, :) − A(i, col) ∗ A(row, :)
    end if
end for
```

Given problem (1) then let d be the direction of the gradient function. Move in direction –d in the feasible space as possible for minimization and move in direction d for maximization. Find the first constraint which meets gradient function. The optimal solution should be located on this this constraint. To find the constraint, move from the feasible solution x_j based on the following equation:

$$x_{j+1} = x_j - \gamma_1 d \tag{6}$$

For minimization and

$$x_{j+1} = x_j + \gamma_1 d \tag{7}$$

For maximization. Continue this process in the feasible region until:

$$d(f(x_{j+1}), f(x_j)) < \epsilon \tag{8}$$

Where d is the following metric and ϵ is a very small positive number.

$$d(f(x_{j+1}), f(x_j)) = \sum_{j=1}^{n} |f(x^i_{j+1}) - f(x^i_j)| \tag{9}$$

When (8) is satisfied then set the slack variable of the constraint identified in Algorithm 1 to zero and continue using Simplex Method to find the optimal solution.

There is no guarantee to provide an optimal solution by only moving along the direction of the gradient function in the feasible region, on the other hand, the simplex algorithm has a high complexity to solve linear programming problems. One of the advantages of combining these two methods in GSA is that the optimal solution is found with complexity much lesser than the simplex algorithm. Also, by this combination, GSA uses both boundary and insides of the feasible region. Finally, GSA moves very fast at the first stage and then provides the optimal solution by the second stage of the algorithm.

III. Evolutionary–Gradient Algorithm (EGA)

The EGA, for NLPs, is explained in detail in this section. EGA applies an evolutionary population to approximate gradient so as to advance iteratively to a better solution. An approximation of the gradient is constructed using the smallest and largest observed objective values drawn among the solutions of the population. The algorithm will converge to a better solution by updating the approximated gradient and the population.

1) Phase 1. Initial population and approximation of gradients: An initial feasible solution is sampled randomly, then an initial population is created in the neibourghood of the initial solution based on the following inequality:

$$||x_i - x_0|| \leq k, i = 1, 2..., m \tag{10}$$

where x_0 indicates the initial feasible solution, x_i the entry of the initial population and m the population size. k is an arbitrary positive real constant which is defined based on the size of the feasible region. With smaller choice of k one can expect better accuracy the solution but lower convergence speed. Among this population, all m feasible solutions are selected and sorted based on the objective function. The vector pointing from the minimum solution to the maximum solution defines an approximated gradient vector.

2) Phase 2. Continue in the direction of the approximated gradient: In this phase, the algorithm moves in the direction of the approximated gradient to find better solutions. The movement for maximization problem is according to

$$x_{new} = \gamma_2(x_{max} - x_{min}) \approx \gamma_2 d \tag{11}$$

where $(x_{max} - x_{min})$ is an approximation of gradient vector, d, and $\gamma_2 \geq 1$ is a real constant which is defined based on the size of feasible region, x_{max} , x_{min} are solutions with maximum and minimum objective values in the current population. In case of minimization problem, the algorithm uses the following equation:

$$x_{new} = \gamma_2(x_{min} - x_{max}) \approx -\gamma_2 d \tag{12}$$

EGA continues with x_{new}. If x_{new} is feasible it becomes the new initial solution, i.e. $x_0 = -x_{new}$ for the next iteration and the algorithm moves back to Phase 1. If x_{new} is infeasible, it means γ_2 is too big so we change its length with $\gamma_2 = \gamma_2/2$ until it finds a γ_2 that fits into the feasible region. The resulting $\gamma_2 d$ (or $-\gamma_2 d$ for minimization) becomes the x_0 in next iteration.

3) Steps of the algorithm:

1) x_0 is created randomly in feasible region.

2) M feasible random solutions are generated in the neighbourhood of x_0.

3) The vector between minimum and maximum of M solutions is calculated as the approximated gradient vector.

4) Move in the direction of approximated gradient (for maximization problem) based on the following equation:

$$x_{new} = \gamma_2(x_{max} - x_{min}) \approx \gamma_2 d \tag{13}$$

where d represents the approximated gradient vector and $\gamma_2 \geq 1$ is defined based on feasible region size.

5) If x_{new} is feasible let $x_0 = -x_{new}$ and go back to step 2.

6) If x_{new} is infeasible let $\gamma_2 = \gamma_2/2$ and $x_{new} = \gamma_2 d$ and go back to step 5.

In order to solve complicated optimization problems such as multi-objective problems, EGA evaluates the solutions of population based on (14).

$$b_s = \begin{cases} x_i & if \ \sum_{k=1}^{n}(f_k(x_i) - f_k(x_j)) < 0 \\ \\ x_j & if \ \sum_{k=1}^{n}(f_k(x_i) - f_k(x_j)) > 0 \end{cases} \tag{14}$$

If all objective functions are minimization problems and b_s is the better solution of x_i, x_j and $f = (f_1, f_1, ..., f_n)$. Also, we have the following equations in the process of EGA for calculate the approximated gradient in (11):

$$x_{min} = \min_{x_i} \sum_{k=1}^{n} f_k(x_i) \tag{15}$$

$$x_{max} = \max_{x_i} \sum_{k=1}^{n} f_k(x_i) \tag{16}$$

Figure 1 shows the process of the EGA for a given optimization problem to find the optimal solutions.

Initial population generated by step 2 are shown in Figure 1(a) (blue points) together with the estimated gradient that is built from the minimum to the maximum of the population. Figure 1(b) shows the advances of the algorithm to the second generation and the new set of population points that are closely nested near the optimal solution. Figure 1(c) shows the final convergence towards the global optimum after one iteration. Algorithm 3 shows the pseudo code of EGA.

Some of the advantages of EGA are as follows:

1) Finding and using gradient functions for nonlinear programming problems is very time-consuming, EGS makes an approximated gradient function by initial population very fast.
2) Approximated gradient function is improved by improving populations.
3) EGA considers gradient function, unlike evolutionary algorithms.
4) EGS improves provided feasible solutions and populations gradually.

IV. Complexity of GSA

The total number of vertices in a LP problem with m constraints and n variables to be explored by the Simplex Algorithm is:

$$N(SA) = \binom{m+n}{n} \tag{17}$$

If solved by GSA the total number of vertices reduces to:

$$N(GSA) = \binom{m+n-1}{n-1} \tag{18}$$

as one of the basic constraint would be identified by the gradient step before applying Simplex Algorithm.

Algorithm 3 Pseudo code of EGA

```
Initialization of Variables:
γ2: A real constant greater than 1
k: An arbitrary positive real constant
M: Number of Iterations
ϵ: A small positive number
x0: Initial solution
n: Number of solutions
Let NI=0
while 1¿0 do
    Generate x0
    if x0 is feasible then
        Break
    end if
end while
for i = 1 to n do
    Generate xi based on ||xi − x0|| ≤ k
end for
Find xmin, xmax
Let xnew = γ2(xmax − xmin)
Let x0 = xbest, NI = NI + 1
if NI < M then
    Go back to 3
end if
```

Therefore, we have:

$$\alpha = \frac{N(SA)}{N(GSA)} = \frac{\binom{m+n}{n}}{\binom{m+n-1}{n-1}} = \frac{\frac{(m+n)(m+n-1)\dots n!}{n!m!}}{\frac{(m+n-1)(m+n-2)\dots(n-1)!}{m!(n-1)!}} = \frac{(m+n)(m+n-1)\dots(n+1)}{(m+n-1)(m+n-2)\dots(n+1)n} = \frac{m+n}{n} = 1 + \frac{m}{n} \tag{19}$$

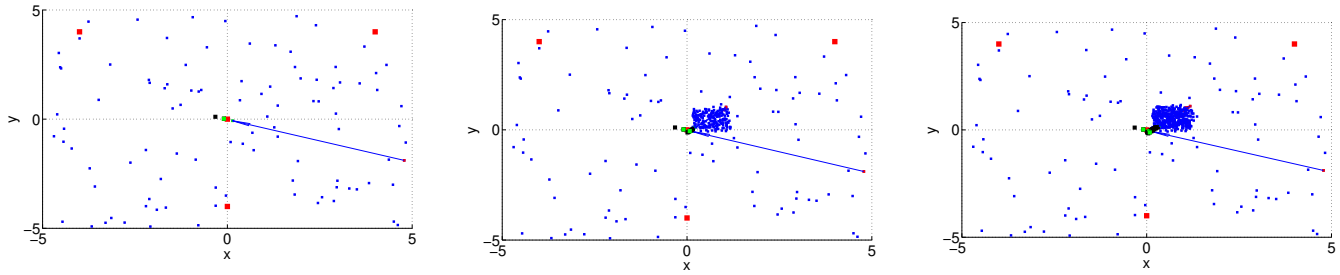
Then:

$$\begin{aligned} \alpha &= \frac{N(SA)}{N(GSA)} \\ m = n &\Rightarrow \alpha = 2 \\ m \ll n &\Rightarrow \alpha \to 1 \\ m \gg n &\Rightarrow \alpha \to \infty \end{aligned} \tag{20}$$

which means that the saving of GSA is more significant when the number of constraints are large. This is also justified in the numerical results.

V. Convergence and Convexity

Convergent factors of EGA include initial solution, direction of gradient function, small positive ϵ (for stopping criteria) and γ (stepsize). EGA can be run several times so as to tweak the convergent parameters of the algorithm and to reveal the convergence rate. If different results are observed, convergence rate is high therefore slow convergent parameter set and very small ϵ are used in this state. If appropriate results are found, convergence rate is common and the usual parameter set will be used. Finally if the same results are found after high number of iterations, the convergence rate is low. In this state a quick convergent parameter set and small ϵ will be used. According to the computational results, convergence rate of EGA is better than simplex method.



(a) Generation 1 and the vector from minimum to maximum solution

(b) Generation 2 close to global optimal

(c) Surrounding optimal solution by EGA

Fig. 1. Steps of EGA for a given optimization problem with four global and local optimals.

For a bounded feasible region EGA method can reach to the constraint, but for unbounded feasible region or very large regions following theorem proves that the used sequence in heuristic algorithm is convergent.

Theorem 1: F_k, sequence of objective function at point x_k in EGA, is convergent to the optimal solution, therefore the algorithm is convergent.

Proof: Let $(f_i) = (f(t^i)) = (f(t_1^i), f(t_2^i), ..., f(t_n^i)) = (f_1^i, f_2^i, ..., f_n^i)$
According to step 4

$$d(f_{j+1}, f_j) = d(f(x_{j+1}), f(x_j)) = \sum_{i=1}^{n} |f(x_{j+1}^i) - f(x_j^i)| < \epsilon$$

So $\sum_{i=1}^{n} |f(x_{j+1}^i) - f(x_j^i)| < \epsilon$, then $|f(x_{j+1}^i) - f(x_j^i)| < \epsilon$ for each i.
Consider sufficient big number like N such that $k+1 > k > N$ and $j = 1, 2, ..., n.$.
we have:

$$|f_{k+1}^i, f_k^i| < \epsilon$$

Now consider m=k+1, r=k thus we have:

$$|f_m^i, f_r^i| < \epsilon \quad For \quad m > r > n$$

The above inequality shows that for each fixed i, $(1 \le i \le n)$, the sequence $(f_1^i, f_2^i, ...)$ is Cauchy of real numbers, thus it converges. Now let $f_m^i \to f^i$ when $m \to \infty$. Using all of these n limits, $f = (f^1, f^2, ..., f^n)$. Using (6) and m=k+1, r=k,

$$d(f_m, f_r) < \epsilon$$

Finally if $r \to \infty$, by $f_r \to f$ then

$$d(f_m, f) \le \epsilon$$

The last inequality shows f is the limit of (f_m) and therefore the sequence is convergent. GSA has two phases; phase 1 is a heuristic method which moves in direction of the gradient function inside feasible region and phase 2 in which the obtained problem from phase 1 will be solved by the simplex method. In this section, at first a discussion of the convergence of the heuristic algorithm in phase 1 is presented and then the convexity of the obtained problem from phase 1 is discussed. As mentioned the algorithm in phase 2 is the the simplex method. GSA is convergent for bounded and convex mathematical programming problems. Now consider GSA which uses the simplex method to solve the obtained problem from phase1. Therefore only convexity of the obtained problem from phase 1 is required. If the k-th constraint was selected from the heuristic algorithm in phase 1, this constraint combined with problem (1) construct the following problem:

$$\begin{aligned} &\min cx \\ &s.t \\ &\sum_{j=1}^{n} a_{kj} x_j = s_k \end{aligned} \tag{21}$$

Problem (7) is convex because:

$$\alpha \sum_{j=1}^{n} a_{kj} x_j = \alpha s_k \tag{22}$$

And

$$\beta \sum_{j=1}^{n} a_{kj} y_j = \beta s_k \tag{23}$$

From (19), (20)

$$\begin{aligned} &\sum_{j=1}^{n} a_{kj} \alpha x_j + \sum_{j=1}^{n} a_{kj} \beta y_j \\ &= \alpha \sum_{j=1}^{n} a_{kj} x_j + \beta \sum_{j=1}^{n} a_{kj} y_j \\ &= \alpha s_k + \beta s_k = (\alpha + \beta) s_k \end{aligned} \tag{24}$$

Which this finishes proof of the convexity.

VI. Computational Results

In this section, both LP and NLP in both categories of small and large sizes are solved to demonstrate the effectiveness of the designed algorithm.

A. GSA for LPs

Two examples will be solved by the GSA.

The optimal solution of Example 1 in Table I is:

$$x_1^* = 2, x_2^* = 2/3, z^* = 8/3 \tag{25}$$

TABLE I. Optimization test functions examples 1-8

Examples	Equation
Example 1	$\max x_1 + x_2 s.t : 2x_1 + 3x_2 \leq 6, -x_1 + x_2 \leq 1, x_1 \leq 2, x_1, x_2 \geq 0.$
Example 2	$\min x_1 + x_2 - 4x_3 s.t : x_1 + x_2 + 2x_3 \leq 9, x_1 + x_2 - x_3 \leq 2, -x_1 + x_2 + x_3 \leq 4, x_1, x_2, x_3 \geq 0.$
Example 3	$f(x,y) = e^{-(x-4)^2-(y-4)^2} + e^{-(x+4)^2-(y-4)^2} + 2e^{-x^2-y^2} + 2e^{-x^2-(y+4)^2}$
Example 4 - Rastrigin Function	$f(x,y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
Example 5	$f(x,y) = -20e^{-0.2\sqrt{(0.5(x^2+y^2))}} + e^{0.5(cos(2\pi x)+cos(2\pi y))} + e + 20$
Example 6 - Holder-Table Function	$f(x,y) = -\lvert sin(x)cos(y)e^{\lvert 1-\frac{\sqrt{x^2+y^2}}{\pi}\rvert}\rvert$
Example 7 - Rosenbrock Function	$f(x,y) = \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$
Example 8 - Bird Function	$f(x,y) = sin(x)e^{(1-cos(y))^2} + cos(y)e^{(1-sin(x))^2} + (x-y)^2$
Example 9 - Salomon Function	$f(\mathbf{x}) = f(x_1, ..., x_n) = 1 - cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$
Example 10 - Keane Function	$f(x,y) = -\frac{\sin^2(x-y)\sin^2(x+y)}{\sqrt{x^2+y^2}}$
Example 11 - Bohachevsky N. 2 Function	$f(x,y) = x^2 + 2y^2 - 0.3cos(3\pi x)cos(4\pi y) + 0.3$

To move in the direction of the gradient vector, let $X = (x_1, x_2) = \alpha(1,1)$. Then the following constraints will be obtained:

$$\begin{aligned} 2\alpha + 3\alpha &\leq 6 \\ -\alpha + \alpha &\leq 1 \\ \alpha &\leq 2 \end{aligned} \tag{26}$$

Therefore

$$\begin{aligned} 5\alpha &\leq 6 \\ 0 &\leq 1 \\ \alpha &\leq 2 \end{aligned} \tag{27}$$

The first constraint is selected to exit from basic in simplex. Then we continue to solve the problem using simplex algorithm as follows.

	z	x_1	x_2	s_1	s_2	s_3	RHS
z	1	−1	−1	0	0	0	0
s_1	0	2	3	1	0	0	6
s_2	0	−1	1	0	1	0	1
s_3	0	1	0	0	0	1	2

(28)

Iteration 2:

	z	x_1	x_2	s_1	s_2	s_3	RHS
z	1	0	1/2	1/2	0	0	3
x_1	0	1	3/2	1/2	0	0	3
s_2	0	0	5/2	1/2	1	0	4
s_3	0	0	−3/2	−1/2	0	1	−1

(29)

Iteration 3:

	z	x_1	x_2	s_1	s_2	s_3	RHS
z	1	0	0	1/3	0	1/3	8/3
x_1	0	1	0	0	0	1	2
s_2	0	0	0	−1/3	1	5/3	7/3
x_2	0	0	1	1/3	0	−2/3	2/3

(30)

Figure 2 (a-b) shows the solution process of the algorithm for Example 1. The algorithm starts from origin searching in direction of the gradient of objective function (blue vector) until one constraint is found (Figure 2(a)). After that in the second phase, the algorithm moves from the green point found in phase 1 into a vertex on the identified constraint (Figure 2(b)), which is the optimal solution.

Optimal solution of Example 2 in Table I as follows:

$$x_1^* = 1/3, x_2^* = 0, x_3^* = 13/3, z^* = -17 \tag{31}$$

To move in direction of the gradient vector, let $X = (x_1, x_2, x_3) = -\alpha(1,1,-4)$. Then the following constraints will be obtained:

$$\begin{aligned} -\alpha - \alpha + 8\alpha &\leq 9 \\ -\alpha - \alpha - 4\alpha &\leq 2 \\ -\alpha - \alpha + 4\alpha &\leq 4 \end{aligned} \tag{32}$$

Therefore

$$\begin{aligned} 6\alpha &\leq 9 \\ -6\alpha &\leq 2 \\ 4\alpha &\leq 4 \end{aligned} \tag{33}$$

So, the third constraint is selected to exit from basic in simplex. Continue with simplex algorithm:

	z	x_1	x_2	x_3	s_1	s_2	s_3	b
z	1	−1	−1	4	0	0	0	0
s_1	0	1	1	2	1	0	0	9
s_2	0	1	1	−1	0	1	0	2
s_3	0	−1	1	1	0	0	1	4

(34)

Iteration 2:

	z	x_1	x_2	x_3	s_1	s_2	s_3	b
z	1	3	−5	0	0	0	−4	−16
s_1	0	3	−1	0	1	0	−2	1
s_2	0	0	2	0	0	1	1	6
x_3	0	−1	1	1	0	0	1	4

(35)

Iteration 3:

	z	x_1	x_2	x_3	s_1	s_2	s_3	b
z	1	0	−4	0	−1	0	−2	−17
x_1	0	1	−1/3	0	1/3	0	−2/3	1/3
s_2	0	0	2	0	0	1	1	6
x_3	0	0	2/3	1	1/3	0	1/3	13/3

(36)

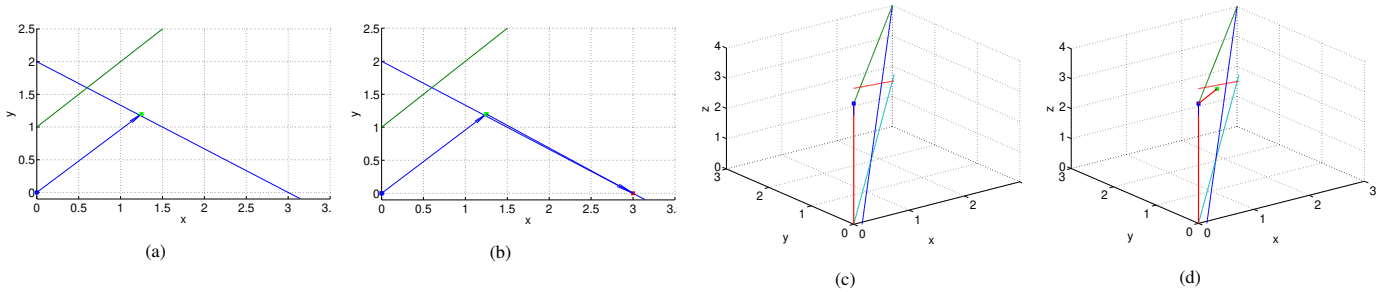


Fig. 2. Steps of GSA for Examples 1, 2.

TABLE II. Comparisons between GSA and normal method in Matlab (Linprog interior point)

Name	size m×n	Optimal	Linprog	GSA	N. Iter
agg	489×163	-3.59E+07	-3.92e+16	-8.19e+07	5
SC50A	51×48	-6.45E+01	-6.53e+20	-64.57	5
AFIRO	28×32	-4.64E+02	-1.45e+29	-464.75	5
SC50B	51×48	-7.00E+01	-3.25e+27	-70.00	5
SHARE2B	28×32	-4.15E+02	-1.93e+19	-549.52	5
Random Problem	2000×4000	———	189.12e+12	188.14e+12	100

Benchmark Functions:

F1:Sphere Function:$\sum_{i=1}^{n} x_i^2$

F2:Schwefel 2.22: $\sum_{i=0}^{n} |x_i| + \prod_{i=0}^{n} |x_i|$

F3:Schwefel 2.21:$\max_i\{|x_i|, 1 \le i \le n\}$

F4:Rosenbrock Function:$\sum_{i=1}^{n-1}(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$

F5:Zakharov Function:$\sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$

F6:Quartic Function:$\sum_{i=0}^{n} ix_i^4 + \text{random}[0, 1)$

F7:Schwefel Function:$\sum_{i=1}^{n}(-x_i sin(\sqrt{|x_i|})) + \alpha \cdot n$

F8:Rastrigin Function:10n + $\sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))\}$

F9:Ackley Function:-20 exp(-0.2 $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$) + 20 + e

F10:Levi N. 13:$\sum_{i=1}^{n}(sin^2(3\pi x_i) + (1 - x_i)^2(1 + sin^2(3\pi x_i))$.

TABLE III. Optimization test functions examples 1-8

Functions	VEA		EGA		GWO		PSO	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
F1	1.73	0.58	1.53	0.58	2319	1237	3.55	2.85
F2	5.12	3.26	5.01	3.13	14.43	5.92	8.71	4.92
F3	1.16	1.26	1.12	1.25	13.09	11.34	21.51	6.71
F4	945.1	813.6	1025	743.7	3425	3304	1132	1357
F5	1.18	0.52	1.12	0.56	5009	3028	86.62	147.3
F6	0.014	0.011	0.011	0.019	0.40	0.11	0.57	0.31
F7	-884	567.3	-853	643.1	-10.7	1162	-672	1352
F8	12.47	8.32	11.04	5.36	89.13	37.95	99.83	24.62
F9	1.32	0.82	1.09	1.15	9.45	3.46	4.29	1.30
F10	0.61	0.10	0.45	0.10	3200	6746	13.38	8.96

Solution process for Example 2 is demonstrated in Figure 2 (c-d). The algorithm begins in direction of the gradient of objective function (red vector) until one constraint is found. Then the algorithm begins from the origin and goes to a vertex in the found constraint (blue point – infeasible solution, Figure 2 (c)). Finally, the algorithm tries to active the constraint and goes to the next vertex point (green point – optimal solution, Figure 2 (d)).

Table I shows the models used for the benchmarks and Figure 3 shows the process of EGA for those benchmarks. Figures 4 shows contours of the benchmark functions and the process of EGSA for solving them. Also, comparisons between GSA and normal method to solve linear programming problems in Matlab (Linprog- interior point or simplex algorithm) are shown in Table II on benchmark problems, the last row of the Table II includes a random generated problem which has not optimal solution.

B. Route Optimization Design in Internet of Vehicles environment

Based on the objective function, the problem maximizes both of the connection quality and connectivity probability of the current routes from origin to destination [13]. The constraints are Signal to Interference and Noise Ratio threshold($SINR_t h$) for finding more trustworthy and conjunct route. Volcano Eruption Algorithm (VEA) [18] has been implemented for finding optimal route from origin to destination. Table IV shows the comparison of results and the rate of improvement for GSA and VEA. An initial solution is generated randomly for both GSA and VEA algorithms. Initial solutions by GSA and VEA will be improved by the process of algorithms, this improvement has been shown after five iterations.

Problem 1:

$$\max_{\zeta} \quad F(\zeta) = \lambda_1 \times PC(\zeta) + \lambda_2 \times SINR(\zeta) \tag{37}$$

where $PC(\zeta) = \prod_{i=1}^{n} PC(e_i)$,

$$SINR(\zeta) = \frac{\sum_{i=1}^{n} SINR(e_i) - \sum_{i=1}^{n} SINR_{th}(e_i))}{\sum_{i=1}^{n} SINR(e_i)}, \tag{38}$$

subject to

$$SINR(\zeta) \ge SINR_{th}(\zeta). \tag{39}$$

In the above problem, F(ζ) is the objective function includes a set of routes Z from origin to destination. $\lambda - 1$ and $\lambda - 2$ are the weights in the simulation which experimentally set and also their summation is equal to 1. PC(ζ) and SINR(ζ)

TABLE IV. Comparison and Improvement from Random Initial Solutions (RIS) by GSA and VEA for internet of vehicles problem 1

Problems	Size n×m	Best Solution by GSA	Best Solution by VEA	Improvement of RIS by GSA	Improvement of RIS by VEA
IoV 1-1	100×100	1000.5450	917.3405	0.251	0.221
IoV 2-1	200×200	1.4329e+04	1.3014e+04	0.328	0.318
IoV 3-1	500×500	6.8147e+04	6.9372e+04	0.188	0.202
IoV 4-1	1000×1000	2.9921e+05	2.8461e+05	0.102	0.097
IoV 5-1	2000×2000	1.4822e+06	1.2831e+06	0.3122	0.303
IoV 6-1	5000×5000	6.8766e+06	6.6281e+06	0.087	0.064
IoV 7-1	10000×10000	3.1450e+07	2.7145e+07	0.088	0.081
IoV 8-1	30000×30000	3.8642e+09	3.7916e+09	0.055	0.053

TABLE V. Comparison of EGA and other algorithms for internet of vehicles problem 2

Problems	Size n×m	LCA	VEA	MVA	EGA
IoV 1-2	10×10	6.1257	**7.1234**	6.0327	6.7456
IoV 2-2	50×50	17.4799	**20.8234**	18.1241	19.0521
IoV 3-2	100×100	45.2681	51.4065	43.2356	**53.1578**
IoV 4-2	200×200	88.4489	100.1293	97.2013	**102.0362**
IoV 5-2	500×500	**226.7274**	213.1343	205.2120	201.1680
IoV 6-2	1000×1000	422.4528	675.3421	**734.1243**	532.0134
IoV 7-2	2000×2000	844.9100	1.2543e+03	1.5432e+03	**1.5621e+03**
IoV 8-2	5000×5000	2.2489e+03	3.1437e+03	**4.2516e+03**	4.1026e+03

are the connectivity and reliability of routes. $PC(e_i)$ and $SINR(e_i)$ show the street's connectivity and link reliability.

Problem 2 [14]:

$$\max_y F(y) = \lambda_1 PC(y) + \lambda_2 PDR(y) + \lambda_3 \frac{D_{th} - D_y}{D(y)} \times \frac{1}{(1 + Dv(y))} \tag{40}$$

$$\text{where} \quad PC(y) = \prod_{i=1}^{n} PC(e_i),$$
$$PDR(y) = \prod_{i=1}^{n} PDR(e_i),$$
$$D(y) = \sum_{i=1}^{n} D(e_i), \tag{41}$$
$$Dv(y) = \sum_{i=1}^{n} \frac{Dv(e_i)}{n},$$

$$\text{subject to} \quad D(y) \le D_{th}. \tag{42}$$

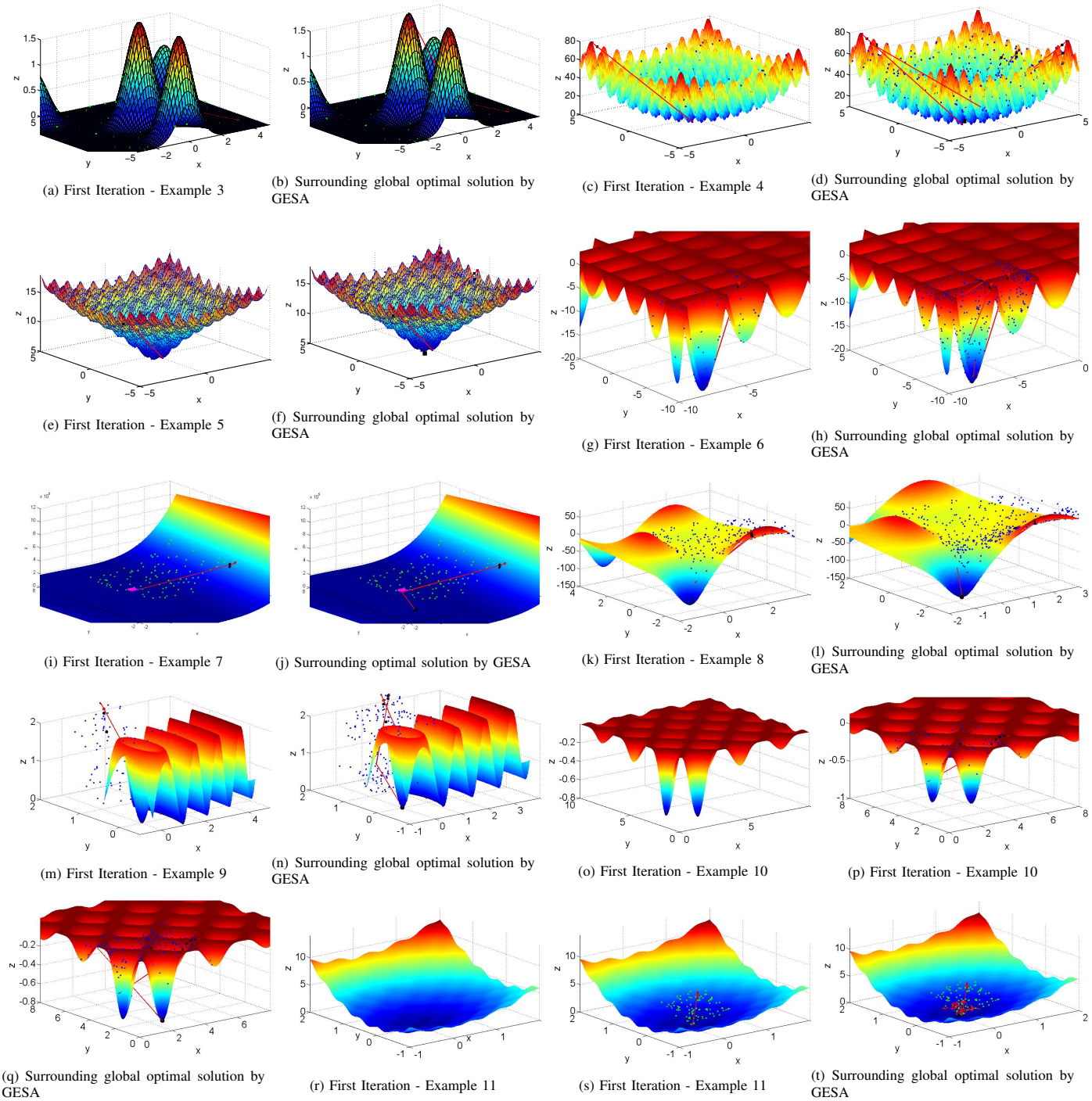
The algorithm is also used to solve internet of vehicles in problem 2. The results have been shown in Table V. The comparison is with some recent efficient meta-heuristics; Laying Chicken Algorithm (LCA), Volcano Eruption Algorithm (VEA), and Multiverse Algorithm (MVA) where GSA has an acceptable results.

C. EGA on NLPs

The number of solutions is 50, the greatest number of iterations is considered 20 and epsilon is 0.1, finally the algorithm is run 30 times. Based on Table III, the obtained results of the proposed algorithm shows that EGA is provided very competitive and impressive results in solving several kinds of test functions. Average results and corresponding standard deviations have been shown by Ave. and Std. respectively. Low standard deviation of EGA is remarkable. It demonstrates that, based on the definition of standard deviation, the values tend to be close to the expected value of the set. The comparison is with some efficient meta-heuristics; Volcano Eruption Algorithm (VEA), Grey Wolf (GWO), and Particle Swarm Optimization (PSO).

VII. Conclusion

EGA and GSA are methods that can optimize linear programming problems and at the same time, the algorithms are also efficient at solving nonlinear programming problems, because they are based on gradient direction. The best solution by the algorithms is better than other methods while using less time. In fact, GSA is successful because it moves infeasible region not just on the boundary. For large problems, this algorithm should be much better than the simplex method because it does not investigate most of the vertex points in the problem. Further testing of this algorithm should be done with big data algorithms as it shows clear potential in this area. Finally, there are still a lot of complicated optimization problems such as quadratic programming, which can be solved by EGA and GSA. Perhaps the easy MATLAB code of the EGA can be a basis for researchers in future research within the area solving problems of large sizes. However the found solution by EGA is only close to the optimal solution an may not be optimal.



(a) First Iteration - Example 3

(b) Surrounding global optimal solution by GESA

(c) First Iteration - Example 4

(d) Surrounding global optimal solution by GESA

(e) First Iteration - Example 5

(f) Surrounding global optimal solution by GESA

(g) First Iteration - Example 6

(h) Surrounding global optimal solution by GESA

(i) First Iteration - Example 7

(j) Surrounding optimal solution by GESA

(k) First Iteration - Example 8

(l) Surrounding global optimal solution by GESA

(m) First Iteration - Example 9

(n) Surrounding global optimal solution by GESA

(o) First Iteration - Example 10

(p) First Iteration - Example 10

(q) Surrounding global optimal solution by GESA

(r) First Iteration - Example 11

(s) First Iteration - Example 11

(t) Surrounding global optimal solution by GESA

Fig. 3. The results of benchmarks when original function (First) and EGA are applied (Others)

Nomenclature

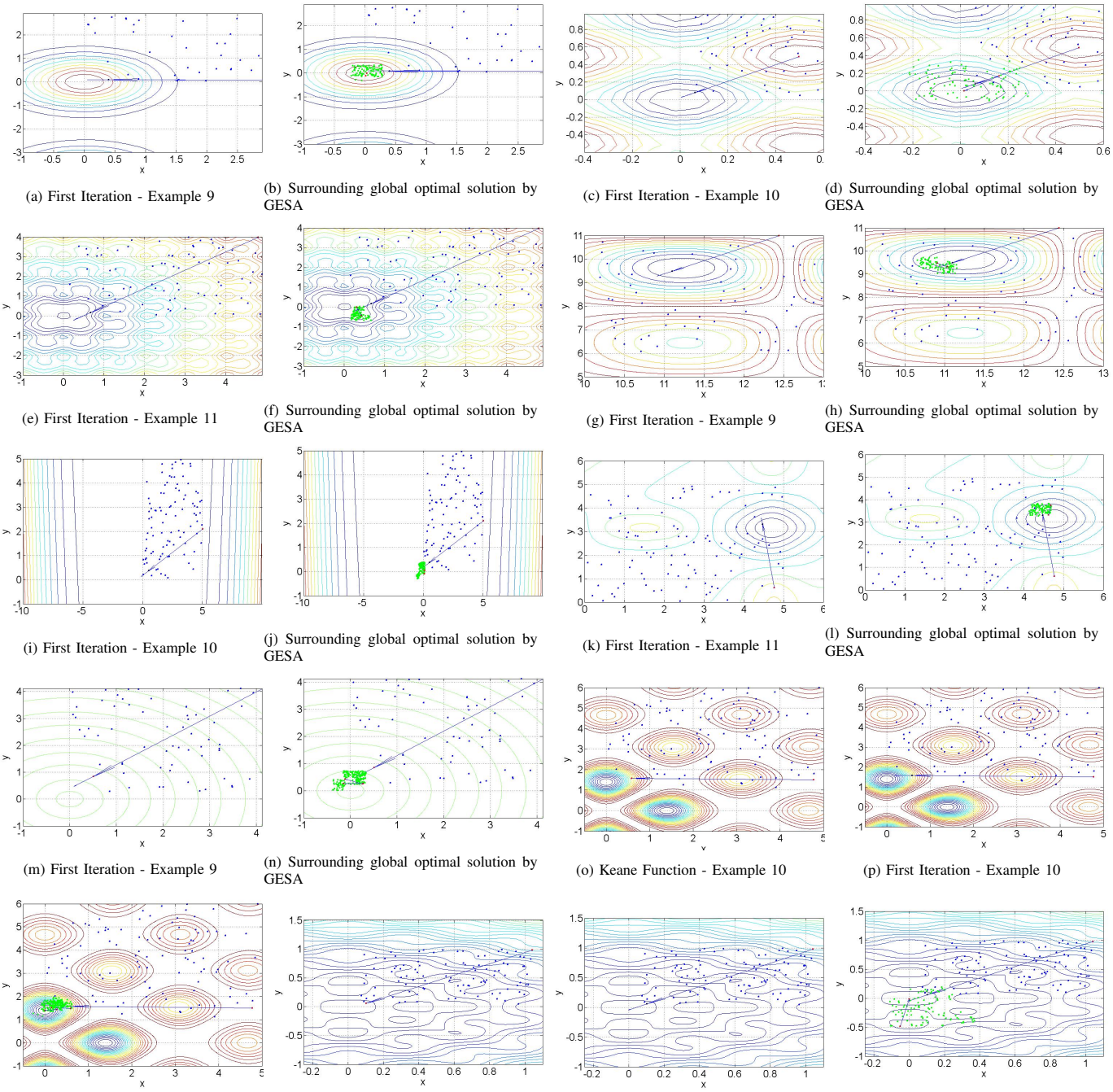
N(GSA): Number of visited solutions by GSA
N(SA): Number of visited solutions by SA
m: Size of population in EGA
k: An arbitrary positive real constant
d: Approximated gradient vector
SA: Simplex Algorithm, MVA: Multiverse Algorithm
VEA: Volcano Eruption Algorithm
RIS: Random Initial Solutions,LCA: Laying Chicken Algorithm
x_0, x_i : $Initial solution, ith solution of population$
γ_2 : $A real constant greater than 1$
α, β : $Parameters to prove convexity$
γ_1 : $A constant to maximize possible moving in GSA$

References

[1] T. Kitahara and S. Mizuno, "A bound for the number of different basic solutions generated by the simplex method," *Mathematical Programming*, vol. 137, no. 1-2, pp. 579–586, 2013.

[2] P.-Z. Peng, J. Yuan, Z.-J. Wang, Y. Yu, and M. Jiang, "An improved gafsa based on chaos search and modified simplex method," *Lecture Notes in Electrical Engineering*, vol. 336, pp. 133–141, 2015.

[3] J.-Y. Chang, S.-H. Liao, S.-L. Wu, and C.-T. Lin, "A hybrid of cuckoo search and simplex method for fuzzy neural network training," 2015, pp. 13–16.



(a) First Iteration - Example 9

(b) Surrounding global optimal solution by GESA

(c) First Iteration - Example 10

(d) Surrounding global optimal solution by GESA

(e) First Iteration - Example 11

(f) Surrounding global optimal solution by GESA

(g) First Iteration - Example 9

(h) Surrounding global optimal solution by GESA

(i) First Iteration - Example 10

(j) Surrounding global optimal solution by GESA

(k) First Iteration - Example 11

(l) Surrounding global optimal solution by GESA

(m) First Iteration - Example 9

(n) Surrounding global optimal solution by GESA

(o) Keane Function - Example 10

(p) First Iteration - Example 10

(q) Surrounding global optimal solution by GESA

(r) Bohachevsky N. 2 Function - Example 11

(s) First Iteration - Example 11

(t) Surrounding global optimal solution by GESA

Fig. 4. Contours of the function and the process of EGSA

[4] M. Riplinger, M. Krause, A. Louis, and C. Xu, "A new local dimming algorithm based on the simplex method," *Computational Optimization and Applications*, vol. 64, no. 1, pp. 243–263, 2016.

[5] V. Cerdà, J. Cerdà, and A. Idris, "Optimization using the gradient and simplex methods," *Talanta*, vol. 148, pp. 641–648, 2016.

[6] M. Dumaldar, "A theoretical comparison between the simplex method and the basic line search algorithm," *Optimization*, vol. 65, no. 1, pp. 1–7, 2016.

[7] M. Bezerra, Q. dos Santos, A. Santos, C. Novaes, S. Ferreira, and V. de Souza, "Simplex optimization: A tutorial approach and recent applications in analytical chemistry," *Microchemical Journal*, vol. 124, pp. 45–54, 2016.

[8] K.-N. Lau, P.-L. Leung, and K.-K. Tse, "Mathematical programming approach to clusterwise regression model and its extensions," *European Journal of Operational Research*, vol. 116, no. 3, pp. 640–652, 1999.

[9] Z. Sener and E. Karsak, "A decision model for setting target levels in quality function deployment using nonlinear programming-based fuzzy regression and optimization," *International Journal of Advanced Manufacturing Technology*, vol. 48, no. 9-12, pp. 1173–1184, 2010.

[10] O. Kocadagli, "A novel nonlinear programming approach for estimating capm beta of an asset using fuzzy regression," *Expert Systems with Applications*, vol. 40, no. 3, pp. 858–865, 2013.

[11] M. Branda, M. Bucher, M. Červinka, and A. Schwartz, "Convergence of a scholtes-type regularization method for cardinality-constrained optimization problems with an application in sparse robust portfolio optimization," *Computational Optimization and Applications*, vol. 70, no. 2, pp. 503–530, 2018.

[12] M. A. Duran and I. E. Grossmann, "Mixed-integer nonlinear

programming algorithm for process synthesis." 1984.
[13] K. Ghafoor, L. Kong, D. Rawat, E. Hosseini, and A. Sadiq, "Quality of service aware routing protocol in software-defined internet of vehicles," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2817–2828, 2019.
[14] G. Li, L. Boukhatem, and J. Wu, "Adaptive quality-of-service-based routing for vehicular ad hoc networks with ant colony optimization," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3249–3264, 2017.
[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," vol. 4, 1995, pp. 1942–1948.
[16] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.
[17] A. Gandomi and A. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.
[18] E. Hosseini, A. Sadiq, K. Ghafoor, D. Rawat, M. Saif, and X. Yang, "Volcano eruption algorithm for solving optimization problems," *Neural Computing and Applications*, vol. 33, no. 7, pp. 2321–2337, 2021.
[19] E. Hosseini, K. Ghafoor, A. Emrouznejad, A. Sadiq, and D. Rawat, "Novel metaheuristic based on multiverse theory for optimization problems in emerging systems," *Applied Intelligence*, vol. 51, no. 6, pp. 3275–3292, 2021.
[20] E. Hosseini, K. Ghafoor, A. Sadiq, M. Guizani, and A. Emrouznejad, "Covid-19 optimizer algorithm, modeling and controlling of coronavirus distribution process," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2765–2775, 2020.
[21] S. Mirjalili, S. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
[22] X.-S. Yang, "Bat algorithm: Literature review and applications," *International Journal of Bio-Inspired Computation*, vol. 5, no. 3, pp. 141–149, 2013.
[23] E. Cuevas, M. Cienfuegos, D. Zaldívar, and M. Pérez-Cisneros, "A swarm optimization algorithm inspired in the behavior of the social-spider," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6374–6384, 2013.
[24] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: Chicken swarm optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8794, pp. 86–94, 2014.
[25] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, 1st ed. NLD: Elsevier Science Publishers B. V., 2014.
[26] S. S. Sefati and N. J. Navimipour, "A qos-aware service composition mechanism in the internet of things using a hidden markov model-based optimization algorithm," *IEEE Internet of Things Journal*, 2021.
[27] A. Forestiero and G. Papuzzo, "Agents-based algorithm for a distributed information system in internet of things," *IEEE Internet of Things Journal*, 2021.
[28] H. Zhang, L. Tong, J. Yu, and J. Lin, "Blockchain aided privacy-preserving outsourcing algorithms of bilinear pairings for internet of things devices," *IEEE Internet of Things Journal*, 2021.

Eghbal Hosseini is currently working on RoRo Green Project as a researcher with Roskilde University, before that, he was working as a senior researcher with Erbil Polytechnic University, and an assistant professor at University of Raparin from 2017 to 2021. He received B.Sc. and M.Sc degrees in applied mathematics and operations research in Iran, and also his Ph.D. degree in optimization from Tehran Payame Noor University at 2015. His research interests are meta-heuristic approaches, algorithms, multilevel programming problems and machine learning. From 2017 he has proposed five new meta-heuristics: Laying Chicken Algorithm (LCA), Big Bang Algorithm (BBA), Volcano Eruption Algorithm (VEA), Multiverse Algorithm (MVA), and Covid-19 Optimizer Algorithm (CVA).

Line Reinhardt is an Associate Professor in the Programming, Logic and Intelligent Systems Section at the Department of People and Technology at Roskilde University and the head of the Center for Maritime and Marine Research at the University. Line Reinhardt has served many years on the board of the Danish Operations Research Society. Line Reinhardt has been in the Organizing committee for several international conferences such as 8th ESICUP conference, 4th ICCL and the 33rd EURO conference. Dr. Reinhardt is engaged in international collaboration among others with Norwegian University of Science and Technology, Paderborn University Germany, Izmir Bakircay University Turkey and Dalian Maritime University in China.

Dr. Danda B. Rawat is a Full Professor in the Department of Electrical Engineering and Computer Science (EECS), Founder and Director of the Howard University Data Science and Cybersecurity Center, Director of DoD Center of Excellence in Artificial Intelligence and Machine Learning (CoE-AIML), Director of Cyber-security and Wireless Networking Innovations (CWiNs) Research Lab, Graduate Program Director of Howard CS Graduate Programs and Director of Graduate Cybersecurity Certificate Program at Howard University, Washington, DC, USA. Dr. Rawat is engaged in research and teaching in the areas of cybersecurity, machine learning, big data analytics and wireless networking for emerging networked systems including cyber-physical systems, Internet-of-Things, multi domain operations, smart cities, software defined systems and vehicular networks. He has secured over 16 million USD in research funding from the US National Science Foundation (NSF), US Department of Homeland Security (DHS), US National Security Agency (NSA), US Department of Energy, National Nuclear Security Administration (NNSA), DoD and DoD Research Labs, Industry (Microsoft, Intel, etc.) and private Foundations. Dr. Rawat is the recipient of NSF CAREER Award in 2016, Department of Homeland Security (DHS) Scientific Leadership Award in 2017, Provost's Distinguished Service Award 2021, Researcher Exemplar Award 2019 and Graduate Faculty Exemplar Award 2019 from Howard University, the US Air Force Research Laboratory (AFRL) Summer Faculty Visiting Fellowship 2017, Outstanding Research Faculty Award (Award for Excellence in Scholarly Activity) at GSU in 2015, the Best Paper Awards (IEEE CCNC, IEEE ICII, BWCA) and Outstanding PhD Researcher Award in 2009. He has delivered over 30 Keynotes and invited speeches at international conferences and workshops. Dr. Rawat has published over 200 scientific/technical articles and 11 books. He has been serving as an Editor/Guest Editor for over 70 international journals including the Associate Editor of IEEE Transactions of Service Computing, Editor of IEEE Internet of Things Journal, Associate Editor of IEEE Transactions of Network Science and Engineering and Technical Editors of IEEE Network. He has been in Organizing Committees for several IEEE flagship conferences such as IEEE INFOCOM, IEEE CNS, IEEE ICC, IEEE GLOBECOM and so on.