https://doi.org/10.25130/tjps.v29i3.1604

Tikrit Journal of Pure Science

ISSN: 1813 – 1662 (Print) --- E-ISSN: 2415 – 1726 (Online)

Journal Homepage: http://tjps.tu.edu.iq/index.php/j

Two Algorithms for Solving Unconstrained Global Optimization by Auxiliary Function Method

Shehab Ahmed Ibrahem[1], Isam Haider Halil[2], Suaad Madhat Abdullah[2]

[1] *College of Computer Science, University of Kirkuk, Kirkuk, Iraq*

[2]*College of Science, University of Kirkuk, Kirkuk, Iraq*

ARTICLE INFO.

Article history:

-Received:	28 / 12 / 2023
-Received in revised form:	24 / 1 / 2024
-Accepted:	30 / 1 / 2024
-Final Proofreading:	25 / 3 / 2024
-Available online:	25 / 6 / 2024

Keywords: Global optimization, Smoothing technique, Auxiliary function.

Corresponding Author:

Name: Shehab Ahmed Ibrahem

E-mail: mullaiq@uokirkuk.edu.iq

Tel:

©2024 THIS IS AN OPEN ACCESS ARTICLE UNDER THE CC BY LICENSE http://creativecommons.org/licenses/by/4.0/

ABSTRACT

In this paper, we present two algorithms that are designed to solve unconstrained global optimization problems. The first algorithm is introduced for resolving unconstrained optimization problems by dividing a multidimensional problem into partitions of a one-dimensional problem and subsequently identifying a global minimizer for each partition by utilizing an auxiliary function and then using it to find the global minimizer of a multidimensional problem. In the second algorithm, the same auxiliary function is used to find a global minimizer of the same multidimensional problem without partitioning it. Finally, we apply these algorithms to common test problems and compare them to each other to show the efficiency of the algorithms.

خوارزميتان لحل التحسين العالمي غير المقيد باستخدام الوظيفة المساعدة

شهاب احمد ابراهيم[1]، عصام حيدر خليل[2] و سعاد مدحت عبدالله[2]

[1] *قسم علوم الحاسوب ، كلية علوم الحاسوب وتكنولوجيا المعلومات ، جامعة كركوك ، كركوك ، العراق*

[2] *قسم الرياضيات ، كلية العلوم ، جامعة كركوك ، كركوك ، العراق*

الملخص

في هذا البحث، نقدم خوارزميتين لحل مشاكل التحسين العالمي غير المقيدة. يتم تقديم الخوارزمية الأولى لحل مشاكل التحسين غير المقيدة عن طريق تقسيم المشكلة متعددة الأبعاد إلى أقسام لمشكلة ذات بعد واحد ومن ثم تحديد المصغر العالمي لكل قسم من خلال استخدام دالة مساعدة ثم استخدامها للعثور على المصغر العالمي للمشكلة متعددة الأبعاد . في الخوارزمية الثانية، يتم استخدام نفس الدالة المساعدة للعثور على المُصغر العالمي لنفس المشكلة متعددة الأبعاد دون تقسيمها. أخيرا، نطبق هذه الخوارزميات على مشكلات الاختبار الشائعة ومقارنتها ببعضها البعض لإظهار كفاءة الخوارزميتين.

https://doi.org/10.25130/tjps.v29i3.1604

1. Introduction

Optimization is an important field of applied mathematics that searches for optimal values, whether maximizers or minimizers, for a specific function. Global optimization can be used to model many real-life scientific problems, for example, engineering, chemical, physics, astronomy, agriculture, economics, information technology, geography, etc. scientific problems [1, 9]. The formulation of global optimization problems as an objective function varies from one problem to another. If the objective function is convex, then the function is simple, and we only need to find the solution using local optimization or local search through one of the local search methods, but if the objective function is non-convex, then the function is complex, and we need to use global optimization methods to find the solution, which searches the entire domain (rather than a specific region) [1].

The general mathematical description of the global optimization problem can be formatted as follows:

$(P) \quad \min_{x\in\Omega} f(x)$ …(1)

where $\Omega \subset R^n$ is the domain area of the objective function f. Depending on the problem type, global optimization problems are divided into two main types, constrained problems (which use part of the domain Ω as a search region) and unconstrained problems (which use the whole domain Ω as a search region for the objective function). In this study, global optimization of unconstrained types is used as a search area. Many important studies in different fields are modeled as global optimization, [1–14]. This paper presents two algorithms to solve unconstrained global optimization problems. The first algorithm is called the directional search algorithm (DSA), which solves optimization problems by transforming a multidimensional problem into partitions of a one-dimensional problem, finding the global minimizer of the one-dimensional problem, and then using it to find the multidimensional problem. The second algorithm is called the multidimensional search algorithm (MSA), which solves optimization problems by using the same auxiliary function to find the global minimizer of the same multidimensional problem without partitioning it, and then comparing the two algorithms to show which one is better at finding the solutions. This study is organized as follows: The second part addresses the inclusion of some definitions and assumptions necessary to describe the type of problem. The third part presents the two algorithms to solve the unconstrained global optimization problem. Part four summarizes the numerical results of the proposed algorithms. Finally, in part five, conclusions are presented.

2. Preliminaries

The following assumptions need to be defined.

Assumption 1. The function $f(x) \to \infty$ *as* $\|x\| \to \infty$ in other words, the function f is coercive.

Assuming 1, it is assumed that there exists a box Ω that is closed and bounded and contains all the minimizers of f.

Assumption 2. It is necessary to have a finite number of local minimizers for the function f, the set Ω_1 and Ω_2 are defined by $\Omega_1 = \{x \in \Omega \,\vdots\, f(x) \geq f(x_j^*),\ x \neq x_j^*\}$ and $\Omega_2 = \{x \in \Omega \mid f(x) < f(x_j^*)\ x \neq x_j^*\}$.

So, we provide the definitions:

Definition 1 [10]: Let $\Omega \subset R^n$, a point $x^* \in \Omega$ is called a global minimizer of f if $f(x^*) \leq f(x)$ for all $x \in \Omega$, and x^* is referred to as a global maximizer of f if $f(x^*) \geq f(x)$ for all $x \in \Omega$.

Definition 2 [10]: A direction $d \in R^n$ s referred to as a descent direction for a given point $x \in R^n$, if there exists $\delta > 0$ such that

$f(x + \alpha d) < f(x),\ for\ all \quad \alpha \in (0, \delta)$

Definition 3 [10]: Suppose that x_j^* is any local minimizer of the function $f(x)$. Then, the set $B(x_j^*) \subset \Omega$ is commonly referred to as a basin of f at point x_j^* if there is a local minimization method that starts from any point $B(x_j^*)$ finds local minimizer x_j^*. So, there exists a higher basin of x_j^*if any minimizer x_{j+1}^* of $f(x)$ hold the $f(x_{j+1}^*) \geq f(x_j^*)$, and if $f(x_{j+1}^*) < f(x_j^*)$ that means lower basin.

3. Theoretical Part

In this part, we suggest two algorithms with two different ideas to find the global optimal of the same problem. The following auxiliary function, taken from [1], was used to obtain a better solution in both algorithms:

$F_{\beta,\mu}(\mathrm{x}, x_j^*) = A_\beta(x, x_j^*) + \psi(x_j^*, \mu)$, … (2)

where

$A_\beta(x, x_k^*) = \left(f(x) - f(x_j^*)\right)\Gamma(t,\beta) + f(x_j^*)$,

$\Gamma(t,\beta) = 1\Big/\left(1 + e^{(t/\beta)}\right)$, $t = f(x) - f(x_j^*), \beta > 0$

and

$\psi(x_j^*, \mu) = 1/\mu + \|x - x_j^*\|^2$, $0 < \mu \leq 1$.

3.1 Directional Search Algorithm (DSA)

In this algorithm, we use any initial point $x_0 \in \Omega$, and then from this point, and by dividing a multidimensional problem based on directions into one-dimensional partitions, we can use the auxiliary function $F_{\beta,\mu}(\mathrm{x}, x_j^*)$ to find the global minimizer for each partition. that is applied only to the one-dimensional problem, to explain that:

Let x_0 any initial point belong to the search area (domain $\Omega \subset R^n$), d_j is a direction and $j = 1,2, \dots, n$, we can use the line

$g(\alpha) = x_0 + \alpha d_j$,

to construct one-dimensional function

$G(\alpha) = f(x_0 + \alpha d_j)$,

then

$\min_\alpha G(\alpha) = f(x_0 + \alpha d_j)$ ….(3)

https://doi.org/10.25130/tjps.v29i3.1604

To find the $\alpha^*$from equation (3), we use the auxiliary function in (2) after transferring it to a one-dimensional function as follows:

$F_{\beta,\mu}(\alpha,\alpha_j^i) = A_\beta(\alpha,\alpha_j^i) + \psi(\alpha_j^i,\mu)$ …..(4)

That is the global minimizer α_j^* at direction d_j give to a result of a one-dimensional minimization problem. Then, find the point x_j' corresponding to point α_j^*using $x_j' = x_0 + \alpha_j^* d_j$, and use the point x_j' as a starting point to get the corresponding local minimizer x_j^* of $f(x)$ by minimize the objective function $f(x)$. Finally, we get all minimizers $x_j^*$from all directions d_j, $j = 1,2,\dots,n$, and the global minimizer x^* of $f(x)$ is obtained by comparing function values at these local minimizers. We give the steps of the DSA algorithm as follows:

Step1. Locate $j = 0$, $\beta = 0.01$, $\delta = 0.1$, choose appropriate $0 < \mu \le 1$,n the number of directions $d_j\ for\ j = 1,2,\dots,n$, select an initial starting point $x_0 \in \Omega$ and locate boundary of Ω.

Step2. Construct the function $G(\alpha) = f(x_0 + \alpha d_j)$ in a one-dimensional function.

Step3. (1) Identify the local minimum α_j^i of the function $G(\alpha)$ from minimizing any initial starting point α_0. and take $r = -1$.

(2) Construct the auxiliary function $F_{\beta,\mu}(\alpha,\alpha_j^i)$ at α_j^i.

(3) Begin from $\alpha_0 = \alpha_j^i + r\delta$ to find a minimizer α_F of $F_{\beta,\mu}(\alpha,\alpha_j^i)$.

(4) If α_F in Ω go to (5) otherwise goto (7).

(5) Begin from α_F minimize $G(\alpha)$ to obtain lower minimizer α_j^{i+1} and go to

(6) If α_j^{i+1} in Ω take $\alpha_j^i = \alpha_j^{i+1}$ go to (2).

(7) If $r = 1$stop the algorithm, take $\alpha_j^* = \alpha_j^{i+1}$,otherwise take $r = 1$ go to (3).

Step4. Compute the point x_j' from the equation $x_j' = x_0 + \alpha_j^* d_j$, and find the local minimizer x_j^* of the function $f(x)$ by using x_j' as initial point.

Step5. If $j < n$, generate a new search direction $d_j, j = j + 1$, and go to Step 2 otherwise go to Step 6.

Step6. Locate the global minimizer value of the objective function $f(x)$ as $x^* = min\, f(x_1^*), f(x_2^*), \dots, f(x_n^*)\}$.

3.2 Multidimensional Search Algorithm (MSA)

In this algorithm, the global minimizer of the multidimensional objective function is identified without partitioning it in the following manner: Firstly, we select any point x_0 from the domain Ω to locate the minimizer x_j^* of the objective function by employing any local search method. Subsequently, we construct the auxiliary function $F_{\beta,\mu}(\mathrm{x},x_j^*)$ to obtain another local minimizer x_{j+1}^* that is lower than x_j^* and utilize the new point to construct the auxiliary function $F_{\beta,\mu}(\mathrm{x},x_{j+1}^*)$ again at x_{j+1}^* to obtain another lower minimizer. Finally, by repeating the aforementioned process, the global minimizer x^* is identified. We summarize the steps of the MSA algorithm as follows:

Step1. Set $j = 0$, $\beta = 0.01$, $\delta = 0.01$ as a step, give $0 < \mu \le 1$, n the number of directions $d_j\ for\ j = 1,2,\dots,n$, and choose any starting point $x_0 \in \Omega$.

Step2. Minimize the function $f(x)$ by utilizing the initial point x_0, to find a current local minimizer x_j^*.

Step3. Construct the function $F_{\beta,\mu}$ at the local minimizer x_j^*

$F_{\beta,\mu}(\mathrm{x},x_j^*) = A_\beta(x,x_j^*) + \psi(x_j^*,\mu)$.

Step4. If $j \le n$, set $x = x_j^* + \delta d_j$ proceed to step 5; if not, proceed to Step 6.

Step5. Begin by locating a minimizer x_F of $F_{\beta,\mu}(\mathrm{x},x_j^*)$ by using the point x and if $x_F \in \Omega$ then set $x_0 = x_F$, $j = j + 1$ and go to Step 2; otherwise $j = j + 1$ go to Step 4.

Step6. Chose x_j^* as a global minimizer of $f(x)$ and stop the algorithm.

4. Numerical Experiments

In this part, we applied the proposed algorithms to a set of 21 common test functions, taken from [15], with different dimensions ranging from simple to difficult, as shown in Table 1, to illustrate the strengths and weaknesses of these algorithms. For each problem, ten different starting points are used. The percentage of each algorithm reaching the optimal value is shown, as well as the calculation of the time taken and the evaluation of the auxiliary function and the objective function for each algorithm. All results were calculated on a computer with the following specifications: Intel(R) Core(TM) (CPU i7-3687U, 2.60 GHz) on MATLAB version R2016a. The symbols used in the tables can be explained as follows:

- $\boldsymbol{S_B}$: The symbol of the objective functions used in the test
- $\boldsymbol{n}$: Dimensions of different test functions.
- $\boldsymbol{F_E}$: Average number of functions evaluation for auxiliary function and the objective function over ten different points
- $\boldsymbol{Time}$: determine the average of 10 runs' total running time in seconds.
- $\boldsymbol{F_M}$: The average of objective function values found from ten attempts.
- $\boldsymbol{F_B}$: in ten attempts, the best function value.
- $\boldsymbol{R_S}$: The number of successful attempts to reach the optimal solution from ten starting points.

Table 1: Test Problems

Function No.	Function name	Dimension n	Optimum value	Search area
P_1	Two-dimensional function $c = 0.05$	2	$zero$	$[-c, c]^2, c = 3$
P_2	Two-dimensional function $c = 0.2$	2	$zero$	$[-c, c]^2, c = 3$
P_3	Two-dimensional function $c = 0.5$	2	$zero$	$[-c, c]^2, c = 3$
P_4	Three-hump back camel function	2	$zero$	$[-c, c]^2, c = 3$
P_5	Six-hump back camel function	2	-1.0316	$[-c, c]^2, c = 3$
P_6	Treccani function	2	$zero$	$[-c, c]^2, c = 3$
P_7	Goldstein and Price function	2	3.000	$[-c, c]^2, c = 3$
P_8	Two-dimensional Shubert function	2	-186.73091	$[-c, c]^2, c = 10$
P_9	Rastrigin function	2	-2.000	$[-c, c]^2, c = 3$
P_{10}	(RC)Branin function	2	0.3979	$[-5, 10] \times [10, 15]$
P_{11}	$(S_{4,5})$Shekel function	4	-10.1532	$[0, 10]^4$
P_{12}	$(S_{4,7})$Shekel function	4	-10.4029	$[0, 10]^4$
P_{13}	$(S_{4,10})$Shekel function	4	-10.5364	$[0, 10]^4$
P_{14}	(L_2)Levy function	2	$zero$	$[-c, c]^2, c = 10$
P_{15}	(L_3)Levy function	3	$zero$	$[-c, c]^3, c = 10$
P_{16}	(L_5)Levy function	5	$zero$	$[-c, c]^5, c = 10$
P_{17}	(L_7)Levy function	7	$zero$	$[-c, c]^7, c = 10$
P_{18}	(L_{10})Levy function	10	$zero$	$[-c, c]^{10}, c = 10$
P_{19}	(L_{20})Levy function	20	$zero$	$[-c, c]^{20}, c = 10$
P_{20}	(L_{30})Levy function	30	$zero$	$[-c, c]^{30}, c = 10$
P_{21}	(L_{50})Levy function	50	$zero$	$[-c, c]^{50}, c = 10$

Tables 2 and 3 show the results obtained by the DSA and MSA algorithms on test problems P_1–P_{21}. These Tables contain seven columns as follows: problem code, problem dimension, function evaluation, average of objective function, best function value, total running time and successful attempts.

Table 2: Numerical results of the algorithm DSA.

S_B	n	F_E	F_M	F_B	$Time$	R_S
P_1	2	108	2-0985e-15	3.9321e-17	0.0961	10/10
P_2	2	112	1.4907e-14	1.4123e-18	0.2938	10/10
P_3	2	268	2.0892e-14	1.9526e-15	0.1001	10/10
P_4	2	286	4.1917e-12	1.2678e-16	0.0723	10/10
P_5	2	144	-1.0316	-1.0316	0.0939	10/10
P_6	2	198	2.7807e-13	1.5459e-17	0.0156	10/10
P_7	2	188	3.0000	3.0000	0.0893	10/10
P_8	2	352	-186.7309	-186.7309	0.0815	10/10
P_9	2	336	3.3765e-12	2.1316e-14	0.1265	8/10
P_{10}	2	144	0.3979	0.3979	0.0101	10/10
P_{11}	4	468	-10.1532	-10.1532	0.1597	9/10
P_{12}	4	312	-10.4029	-10.4029	0.2785	10/10
P_{13}	4	364	-10.5321	-10.5321	0.01323	10/10
P_{14}	2	204	3.4651e-15	5.5907e-17	0.0761	10/10
P_{15}	3	286	5.3208e-15	1.4839e-16	0.0134	9/10
P_{16}	5	626	2.2361e-12	4.5660e-15	0.0154	8/10
P_{17}	7	728	1.5423e-14	2.1645e-16	0.0179	8/10
P_{18}	10	788	3.9034e-13	4.6800e-16	0.3213	7/10
P_{19}	20	1490	4.7612e-14	1.9578e-15	0.2745	7/10
P_{20}	30	2480	1.5541e-13	1.9207e-16	0.3123	7/10
P_{21}	50	9520	2.4092e-13	4.1172e-15	0.5545	7/10

https://doi.org/10.25130/tjps.v29i3.1604

Table 3: Numerical results of the algorithm MSA.

S_B	n	F_E	F_M	F_B	$Time$	R_S
P_1	2	291	1.9542e-10	5.7244e-16	0.2563	9/10
P_2	2	315	2.2610e-13	1.2548e-14	0.6049	10/10
P_3	2	288	2.2796e-13	5.7321e-15	0.2113	8/10
P_4	2	306	3.3918e-12	2.2390e-16	0.1015	10/10
P_5	2	135	-1.0316	-1.0316	0.1222	10/10
P_6	2	240	4.8822e-10	5.1253e-16	0.0367	10/10
P_7	2	309	3.0000	3.0000	0.1212	8/10
P_8	2	273	-186.7309	-186.7309	0.1609	9/10
P_9	2	510	5.4921e-12	2.1316e-14	0.3910	6/10
P_{10}	2	243	0.3979	0.3979	0.0226	10/10
P_{11}	4	660	-10.1532	-10.1532	0.3048	6/10
P_{12}	4	545	-10.4029	-10.4029	0.5099	6/10
P_{13}	4	755	-10.5321	-10.5321	0.0278	6/10
P_{14}	2	669	5.6126e-14	2.3476e-16	0.0350	8/10
P_{15}	3	625	6.9832e-13	4.5909e-16	0.0227	7/10
P_{16}	5	1422	4.5783e-13	8.6884e-14	0.0331	6/10
P_{17}	7	1936	2.1327e-10	6.9033e-16	0.0388	6/10
P_{18}	10	2343	1.7542e-11	4.2329e-14	0.7221	6/10
P_{19}	20	9114	3.6532e-13	4.5555e-15	0.5435	6/10
P_{20}	30	13391	5.4447e-12	3.4219e-14	0.4026	6/10
P_{21}	50	48450	5.8142e-13	9.8943e-15	1.0644	7/10

Table 4: Comparison of numerical results between the algorithm DSA and the algorithm MSA

		Algorithm **DSA**.					Algorithm **MSA**			
S_B	n	F_E	F_B	$Time$	R_S		F_E	F_B	$Time$	R_S
P_1	2	108	2.0930e-30	0.2563	10/10		291	5.7244e-16	0.2563	9/10
P_2	2	112	5.9220e-32	0.6049	10/10		315	1.2548e-14	0.6049	10/10
P_3	2	268	1.9526e-15	0.2113	10/10		288	5.7321e-15	0.2113	8/10
P_4	2	286	1.2678e-16	0.1015	10/10		306	2.2390e-16	0.1015	10/10
P_5	2	144	-1.0316	0.1222	10/10		135	-1.0316	0.1222	10/10
P_6	2	198	1.5459e-17	0.0367	10/10		240	5.1253e-16	0.0367	10/10
P_7	2	188	3.0000	0.1212	10/10		309	3.0000	0.1212	8/10
P_8	2	352	-186.7309	0.1609	10/10		273	-186.7309	0.1609	9/10
P_9	2	336	2.1316e-14	0.3910	8/10		510	2.1316e-14	0.3910	6/10
P_{10}	2	144	0.3979	0.0226	10/10		243	0.3979	0.0226	10/10
P_{11}	4	468	-10.1532	0.3048	9/10		660	-10.1532	0.3048	6/10
P_{12}	4	312	-10.4029	0.5099	10/10		545	-10.4029	0.5099	6/10
P_{13}	4	364	-10.5321	0.0278	10/10		755	-10.5321	0.0278	6/10
P_{14}	2	204	5.5907e-17	0.0350	10/10		669	2.3476e-16	0.0350	8/10
P_{15}	3	286	1.4839e-16	0.0227	9/10		625	4.5909e-16	0.0227	7/10
P_{16}	5	626	4.5660e-15	0.0331	8/10		1422	8.6884e-14	0.0331	6/10
P_{17}	7	728	2.1645e-16	0.0388	8/10		1936	6.9033e-16	0.0388	6/10
P_{18}	10	788	4.6800e-16	0.7221	7/10		2343	4.2329e-14	0.7221	6/10
P_{19}	20	1490	1.9578e-15	0.5435	7/10		9114	4.5555e-15	0.5435	6/10
P_{20}	30	2480	1.9207e-16	0.4026	7/10		13391	3.4219e-14	0.4026	6/10
P_{21}	50	9520	4.1172e-15	1.0644	7/10		48450	9.8943e-15	1.0644	7/10

By comparing the results of algorithm DSA with algorithm MSA, we see the superiority of algorithm DSA in all results due to dividing the multi-dimension problem into parts of one-dimensional problems, as algorithm DSA requires a lower function evaluation than algorithm MSA for almost all test problems, as can be seen clearly in the column (F_E). Likewise, in column (F_B), algorithm DSA has an advantage in reaching the global minimizer with more effectiveness and accuracy than algorithm MSA. As for the time spent calculating each problem, algorithm DSA took much less time than algorithm MSA to solve the same problem, as can be seen clearly in column ($Time$), and finally in Column

https://doi.org/10.25130/tjps.v29i3.1604

($\boldsymbol{R_S}$), the successful attempts to reach the global minimizer for algorithm DSA were better than algorithm MSA for all test problems. All comparison results are shown in Table 4.

5. Conclusions

This paper introduces two algorithms, DSA and MSA, to solve unconstrained global optimization problems. The DSA algorithm operates by transforming a multidimensional problem into a one-dimensional problem. While the MSA algorithm finds the global minimizer without dividing the problem into partitions with assistance from the auxiliary function. Then we applied the two algorithms to test problems of different dimensions. When comparing them with each other, the computational results showed that the algorithm DSA was better and faster in terms of time, function evaluation, and number of successful attempts in reaching the global minimizer compared to the algorithm MSA, as can be seen from the relevant tables.

References

[1] Sahiner, A. and Ibrahem, S. A. (2019). A new global optimization technique by auxiliary function method in a directional search. Optimization Letters, 13(2): 309-323.

[2] Kara, G.; Özmen, A. and Weber, G. W. (2019). Stability advances in robust portfolio optimization under parallelepiped uncertainty. Central European Journal of Operations Research, 27: 241-261.

[3] Gharehchopogh, F. S. (2022). An improved tunicate swarm algorithm with best-random mutation strategy for global optimization problems. Journal of Bionic Engineering, 19(4): 1177-1202.

[4] Yilmaz, N. and Sahiner, A. (2017). New global optimization method for non-smooth unconstrained continuous optimization. AIP Conference Proceedings, 1863(1): 250002.

[5] Sahiner, A.; Yilmaz, N. and Ibrahem, S. A. (2018). Smoothing Approximations to Nonsmooth Functions. Journal of Multidisciplinary Modeling and Optimization, *1*(2): 69-74.

[6] Naji, A. J.; Ibrahem S. A. and Umar, U. S. (2023). Improved image segmentation method based on optimized higher-order polynomial. International Journal of Nonlinear Analysis and Applications, 14(1), 2701-2715.

[7] Rmaidh, M. I. and Ibrahim, S. A. (2023). A New Method for Solving Image Segmentation Problems using Global Optimization. International Journal of Intelligent Systems and Applications in Engineering, 11(5): 85-92.

[8] Sahiner, A.; Abdulhamid, I. A. M. and Ibrahem, S. A. (2019). A new filled function method with two parameters in a directional search. Journal of Multidisciplinary Modeling and Optimization, 2(1): 34-42.

[9] El-Gindy, T. M.; Salim, M. S. and Ahmed, A. I. (2016). A new filled function method applied to unconstrained global optimization. Applied mathematics and computation, 273: 1246-1256.

[10] Sahiner, A.; Ibrahem, S. A. and Yilmaz, N. (2020). Increasing the Effects of Auxiliary Function by Multiple Extrema in Global Optimization. In Numerical Solutions of Realistic Nonlinear Phenomena, 125-143.

[11] Liu, H.; Xue, S.; Cheng, Y. and Tuo, S. (2022). A New Parameterless Filled Function Method for Global Optimization. Axioms, 11(12): 746.

[12] Jureczka, M. and Ochal, A. (2021). A nonsmooth optimization approach for hemivariational inequalities with applications to contact mechanics. Applied Mathematics & Optimization, 83: 1465-1485.

[13] Hassan, B. A.; Jabbar, H. N. and Laylani, Y. A. (2023). Upscaling Parameters for Conjugate Gradient Method in Unconstrained Optimization. Journal of Interdisciplinary Mathematics, 26(6): 1171-1180.

[14] Pandiya, R.; Widodo, W. and Endrayanto, I. (2021). Non parameter-filled function for global optimization. Applied Mathematics and Computation, 391: 125642.

[15] Hedar, A. Test functions for unconstrained global optimization. (2013). http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm