Local Selection for Heuristic Algorithms as a Factor in Accelerating Optimum Search

Danuta Jama
Institute of Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: Danuta.Jama@polsl.pl

***Abstract*—Increasing the use of heuristic algorithms in practical applications makes it necessary to perform numerous modifications in order to minimize possible disadvantages. Acceleration and increasing precision have become pressing problems of today's theory of applied computer science and optimization theory. In this work, the idea of a factor which minimizes the time to search the solution space by heuristic algorithms is presented. The proposed modification is described and tested for various test functions. The results are presented and discussed in terms of effectiveness of the proposed modifications.**

I. Introduction

Nowadays, IT solutions can be found everywhere from hospitals and factories to our own homes. Computer systems not only replace men at various levels of action, but also support our lives. In hospitals, the use of such solutions allow for faster diagnosis of disease or even prevent their uprising. In large factories, production of goods is done automatically, without the participation of the people due to a more accurate precision and high speed operation. Moreover, such systems have also found applications in our lives, eg.: washing machines or blood pressure monitors.

Each computer activity in their substrates has numerous algorithms, through which it operates in a certain way. Besides the classic algorithms, the application of computational intelligence is becoming increasingly popular because of the ability to obtain accurate results in a short time. Computational intelligence is represented primarily by three branches – fuzzy logic, neural networks and algorithms inspired by natural phenomena. The last group is classified also to optimization theory, which plays an important role in modern information technology. In [1], Dugun et al. showed that use of these algorithms allows for solving optimization problems of construction and in [2], an optimization technique used in order to solve the capacitated vehicle routing problems was shown. Additionally, in [3], the idea of using one of the optimization algorithms to minimize the drug protein integration energy was discussed. Again in [4], Mohandes discussed the problem of the quality of solar radiation received by the Earth's surface and proposed the idea of modeling global solar radiation using neural networks and optimization algorithm.

Due to the numerous applications of optimization algorithms, in this paper we show that the introduction of local selection as a factor of decreasing the time of the algorithm is an effective modification. For testing purposes, some well-known functions were taken and used for the purposes of verification of the proposed technique.

II. Test functions for optimization purposes

Test functions for optimization problems are called multiobjective function, where the determination of a global minimum or maximum value is an unsolvable problem for well-known algorithms for finding extremes. Multiobjective functions are often called artificial landscapes due to the chart, which is often a complicated surface likened to the known landscapes found in nature.

The first function is called Egg holder's function which includes many local extremes - it can cause getting stuck in one of them. The function is defined as

$$f_{Eggholder}(\overline{x}) = -(x_2+47)\sin\left(\sqrt{\left|\frac{x_1}{2}+(x_2+47\right|}\right) - x_1\sin\left(\sqrt{|x_1-x_2-47|}\right), \quad (1)$$

wherein $\overline{x}$ is a point in $3D$ space with spatial coordinates marked as x_1 and x_2. This function has a number of global minimums/maximums, for instance $f_{Eggholder}(\overline{x}) = -959,6418$ obtained in point $\overline{x} = (512; 404,2319)$. Function is shown in Fig. 1.

Easom function is the second function selected in the minimization problem. The function is described by the following equation

$$f_{Easom}(\overline{x}) = \frac{-\cos(x_1)\cos(x_2)}{\exp((x_1-\pi)^2+(x_2-\pi)^2)}, \quad (2)$$

which reaches a global minimum $f_{Easom}(\overline{x}) = -1$ at the point $\overline{x} = (\pi;\pi)$. Function is shown in Fig. 2.

The last function is a Himmelblau's function representing a flattened landscape and described as

$$f_{Himmelblau}(\overline{x}) = (x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2. \quad (3)$$

This function achieves only one global maximum at the point $\overline{x} = (-0,270845; -0,923039)$ equals $f_{Himmelblau}(\overline{x}) = 181.617$. Function is presented in Fig. 3.

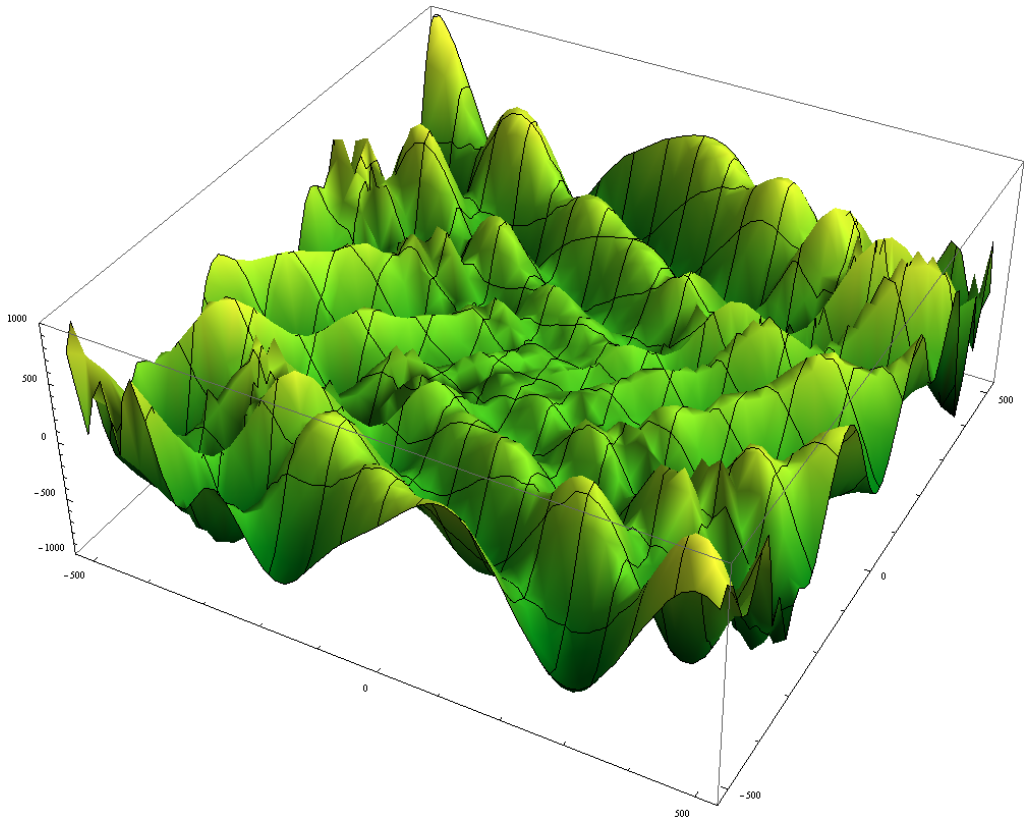

Fig. 1: Egg holder's function

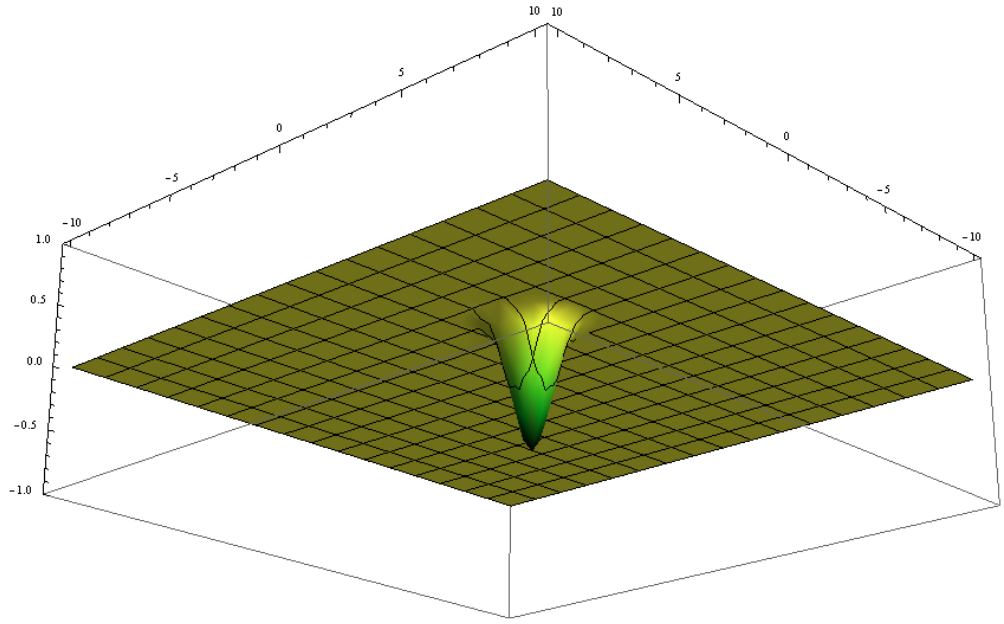

Fig. 2: Easom's function.

III. Heuristic algorithm

Heuristic algorithms are one of the most important techniques for finding the global optimum for multiobjective function in large spaces solutions. Until today, several methods inspired by natural phenomena have been modeled. One of the drawbacks of this type of algorithms is no guarantee of a correct solution at a predetermined number of iterations. The analysis of the convergence of one of the methods, S.Arora and S.Singh shown in [5]. On the other hand, their use allows to

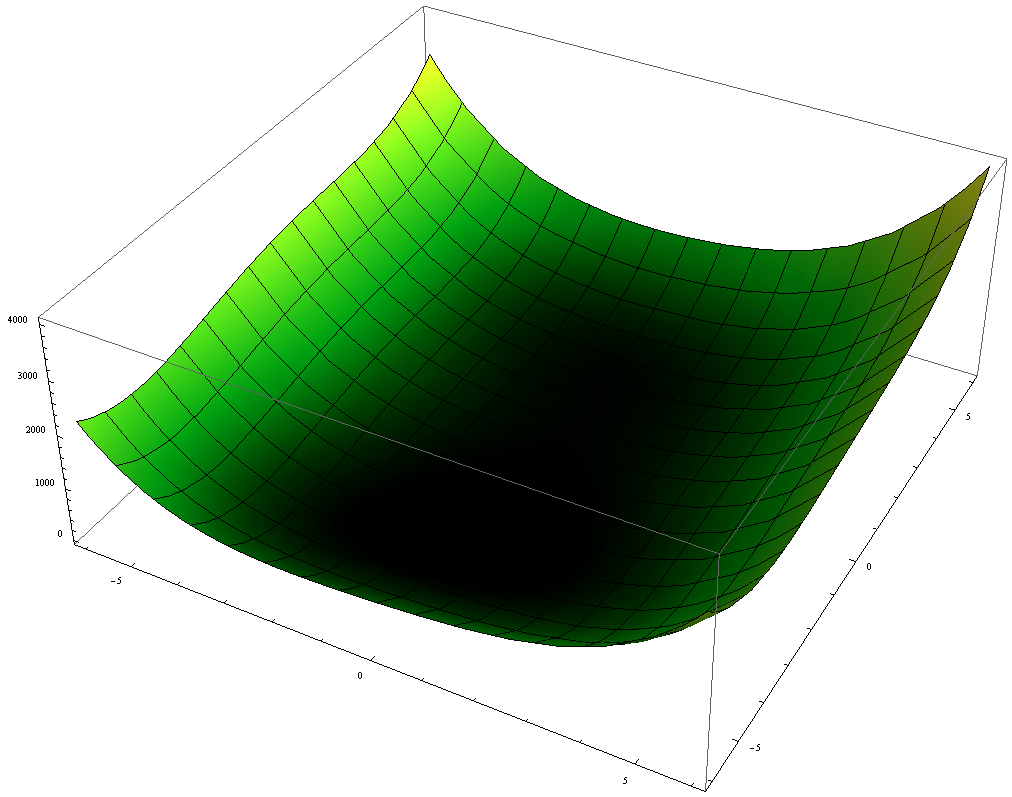

Fig. 3: Himmelblau's function.

obtain in a short time approximate solution using a computer with low computational efficiency.

A. Wolf Search Algorithm

In 2012, Tang et al. [6] have created a model of behavior of wolves while looking for food and avoiding their enemies, which allowed for the creation of bio-inspired heuristic algorithm. The algorithm assumes that the wolf is represented as a point $\overline{x}$ in the solution space. Each wolf has a fixed visual area with a radius r – wolf can sense the company only in the visible range and only in this area, he can move (in one iteration). The position is evaluated in terms of fitness functions $\Theta(\overline{x})$ what is understood as a quality of food in this area. There is a possibility that if the enemy of a wolf appears in his sight, he will escape to another position outside the range of his vision.

Wolves stick together, and so the greater the distance between a pair of wolves, the place is less attractive. Wolf moves to find prey, and therefore a better place (due to the fitness function). The movement carried out by the following equation

$$\overline{x}_{new} = \overline{x}_{actual} + \beta_0 \exp(-r^2)(\overline{x}_{neighbor} - \overline{x}_{actual}) + \gamma, \quad (4)$$

where β_0 means the ultimate incentive (it is a parameter which is set at the beginning of the algorithm), $\overline{x}_{neighbor}$ is the closest neighbor with better adaptation, γ is random number from $[0, 1]$ and r means the distance between $\overline{x}_{actual}$ and $\overline{x}_{neighbor}$ defined as the Euclidean metric

$$r = d(\overline{x}_{actual}, \overline{x}_{neighbor}) = \sqrt{\sum_{i=1}^{2} (\overline{x}_{actual,i} - \overline{x}_{neighbor,i})^2}. \quad (5)$$

Wolves hunt using a stalking process , what can be divided into three stages of preying behavior – initiation, passive action and escape. The first stage involves the movement in the field of view of a wolf, which means finding a better place according to

$$\overline{x}_{new} = \overline{x}_{actual} + \alpha v \gamma, \quad (6)$$

where v is the velocity of a wolf. Passive hunting means stay in your current location, to the moment when the enemy appears. The third stage occurs when the enemy is close to a wolf. Escape is modeled by

$$\overline{x}_{new} = \overline{x}_{actual} + \alpha s \gamma, \quad (7)$$

where s is the step size. Wolf Search Algorithm implementation is presented in Algorithm 1.

B. Water Waves Algorithm

One of the last algorithms inspired by the natural phenomenon algorithm is a model of water waves movement presented by Zheng et al. [7]. A point $\overline{x} = (x_1, x_2)$ in the solution space is called the wave. The algorithm simulates the movement of water waves, and so the following three

Algorithm 1 Wolf Search Algorithm

Start
Define the number of iterations T, number of wolves n, radius of the visual range r, step size s, velocity factor α and coefficient appearance of the enemy p_{alpha}
Define fitness condition $\Theta(\cdot)$
Create an initial population of wolves in random
$t := 0$
while $t \leq T$ **do**
  if $t > 0$ **then**
    Generate 80% of new waves
  end if
  for each wolf $\overline{x}_{actual}$ in population **do**
    Prey initiatively using (6)
    Generate new position $\overline{x}_{new}$ according to (4)
    if $d(\overline{x}_{actual}, \overline{x}_{new}) < r \wedge \Theta(\overline{x}_{new}) > \Theta(\overline{x}_{actual})$ **then**
      Prey new food passively
    end if
    Generate new location
    Generate random number $\beta \in [0, 1]$
    if $\beta > p_{alpha}$ **then**
      Escape to a new position using (7)
    end if
  end for
  if $t \neq T$ **then**
    Take 20% of the best adapted waves to the next iteration
  end if
  $t++$,
end while
Return the best wolf $\overline{x}_{global}$ as a solution
Stop

operations are modeled - propagation, refraction and breaking. Propagation is understood as a movement of waves from deep water to shallow. During this movement, the wave height increases but the length decreases. This step is described as

$$\overline{x}_{new} = \overline{x}_{actual} - 2\lambda_{new}(\overline{x}_{actual})v, \tag{8}$$

where v is random factor in the range of $[-1, 1]$ and $\lambda(y)$ is a function that returns the value of the wavelength y and it is formulated as

$$\lambda_{new}(\overline{y}) = \lambda_{actual}\alpha\frac{\Theta(\overline{y}) - \Theta_{min} + \epsilon}{\Theta_{max} - \Theta_{min} + \epsilon}, \tag{9}$$

where Θ is a fitness function, Θ_{min} and Θ_{max} are respectively the minimum and maximum values in a particular iteration of the algorithm, ϵ is a small number to prevent division by zero and α is the wavelength reduction coefficient.

After propagation, the next stage is refraction – it is a phenomenon of changes in wave direction, where the height reaches a value of 0. The new position after refraction of the wave is modeled as

$$\overline{x}_{actual} = N(\mu, \sigma), \tag{10}$$

where $N(\mu, \sigma)$ is a random number from the normal distribution with μ – mean and σ – standard deviation defined as

$$\begin{cases} \mu = \dfrac{|\overline{x}_{global} + \overline{x}_{actual}|}{2} \\ \sigma = \dfrac{|\overline{x}_{global} - \overline{x}_{actual}|}{2} \end{cases}, \tag{11}$$

where $\overline{x}_{global}$ is the best wave (solution) in current iteration. During refraction, the height of the wave is also modified by

$$\lambda_{new} = \lambda_{old}\frac{\Theta(\overline{x}_{actual})}{\Theta(\overline{x}_{new})}. \tag{12}$$

The final stage of wave movement is breaking, which describes the movement of the wave at a depth below a certain threshold level – the velocity exceeds the wave celerity. The breaking stage is understood as a local search calculated as

$$\overline{x}_{new} = \overline{x}_{actual} + 2\beta N(0, 1), \tag{13}$$

where β is the breaking parameter.

Algorithm 2 Water Waves Algorithm

Start
Define the number of iterations T, number of waves n
Define fitness condition $\Theta(\cdot)$
Create an initial population of waves in random
$t := 0$
while $t \leq T$ **do**
  if $t > 0$ **then**
    Generate 80% of new waves
  end if
  for each $\overline{x}_{actual}$ **do**
    Create $\overline{x}_{new}$ using (8)
    if $f(\overline{x}_{new}) > f(\overline{x}_{actual})$ **then**
      Replace $\overline{x}_{actual}$ with $\overline{x}_{new}$
      if $f(\overline{x}_{new}) > f(\overline{x}_{global})$ **then**
        Break $\overline{x}_{new}$ according to (13)
        Replace $\overline{x}_{global}$ with $\overline{x}_{new}$
      end if
    else
      Reduce the height of the $\overline{x}_{actual}$ by one
      if the height of the wave is 0 **then**
        Refract $\overline{x}_{actual}$ to $\overline{x}_{new}$ by (10)
      end if
    end if
    Update the wavelengths using (9)
  end for
  if $t \neq T$ **then**
    Take 20% of the best adapted waves to the next iteration
  end if
  $t++$,
end while
Return the best wave $\overline{x}_{global}$ as a solution
Stop

IV. A FACTOR IN ACCELERATING THE FINDING OPTIMUM

In the heuristic algorithms, the initial population is selected at random. Such a solution for complex functions increases the risk of getting stuck in a local optimum (see Fig. 1). To remedy this situation, the initial population can be placed in specific locations in the solution space, i.e. all individuals are positioned at equally-spaced. As a result, the population forms a grid that covers the entire space. In the next step, each individual is multiplied to cover a larger area. For every individual is made local search. Each of the individuals is evaluated by fitness function and 25% of the best fittest are selected as the initial population.

Local search is done by a decrease gradient. The algorithm starts at a point $\overline{x}$, for which (the negative for minimization problem) gradient is calculated as

$$\nabla\Theta_{x_i} = \frac{\partial\Theta(x_1, x_2)}{\partial x_i} \qquad \text{for } i = 1, 2. \tag{14}$$

Gradient indicates the direction of the fastest growth the value of the function at the measured point. In the next step, the next point is found according to the direction defined by the gradient as

$$\overline{x}_{new} = \overline{x}_{actual} - \lambda\nabla\Theta_{\overline{x}}, \tag{15}$$

where λ is the value of step. A factor implementation is shown in Algorithm 3.

V. EXPERIMENTAL RESULTS

Numerical tests were performed to search global optimum of all functions described in Section II. For the selected amount of the population ($n = 100$ individuals) and iteration (10, 100, 1000) the results obtained for the original and modified methods. The results in terms of accuracy are presented in Tables I, II and III. The obtained average values for 10 experiments in terms of accuracy of the solution from time are presented in Fig. 4.

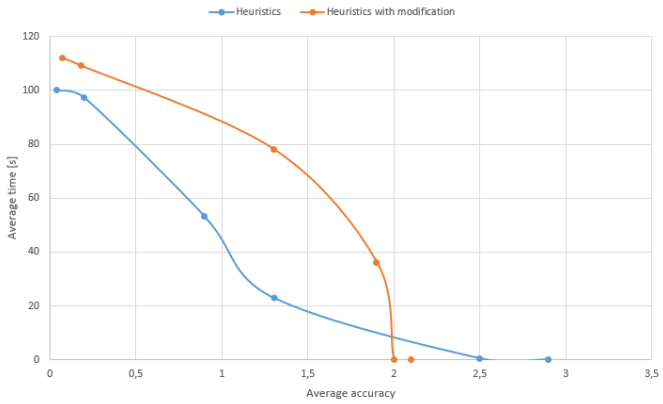

Fig. 4: The dependence of the accuracy from the average time.

Based on the obtained data, the application of the modification in terms of accuracy is only cost-effective for a large number of iterations - the results are more accurate than others. For quantities less than 1000, better accuracy is achieved in the case of original algorithms. In terms of acceleration of the algorithm, additional calculations slow down the operations for small number of iterations. Similar to the accuracy, modifications is valuable when there will be a need for a much larger number of iterations.

VI. FINAL REMARKS

The proposed modification reduces operational time heuristic algorithms and increasing the accuracy of their operation, assuming a large number of iterations or very accurate results. On the basis of the performed test for the selected function, the modification shown that it is possible to reduce the time.

Algorithm 3 A factor algorithm

```
Start
Define
Define fitness function Θ(·), solution space Z × B, population size n, step size λ, number of iteration T
Arrange the n individuals equally-spaced over the entire space AxB
for each individual x̄ do
    Create 3 additional individuals and distribute them near x̄
end for
t = 1,
for each individual x̄ do
    while t ≤ T do
        Calculate x̄_new according to (15)
        if Θ(x̄_actual) ≤ Θ(x̄_new) then
            x̄_old = x̄_new
        end if
        t++
    end while
end for
Rate all the individuals using the fitness function
Return 25% fittest individuals
Stop
```

REFERENCES

[1] I. Durgun and A. R. Yildiz, "Structural design optimization of vehicle components using cuckoo search algorithm," *Materials Testing*, vol. 54, no. 3, pp. 185–188, 2012.

[2] M. Reed, A. Yiannakou, and R. Evering, "An ant colony algorithm for the multi-compartment vehicle routing problem," *Applied Soft Computing*, vol. 15, pp. 169–176, 2014.

[3] A. Ghosh, M. Talukdar, and U. K. Roy, "Stable drug designing by minimizing drug protein interaction energy using pso," *arXiv preprint arXiv:1507.08408*, 2015.

[4] M. A. Mohandes, "Modeling global solar radiation using particle swarm optimization (pso)," *Solar Energy*, vol. 86, no. 11, pp. 3137–3145, 2012.

[5] S. Arora and S. Singh, "The firefly optimization algorithm: convergence analysis and parameter selection," *International Journal of Computer Applications*, vol. 69, no. 3, 2013.

[6] R. Tang, S. Fong, X.-S. Yang, and S. Deb, "Wolf search algorithm with ephemeral memory," in *Digital Information Management (ICDIM), 2012 Seventh International Conference on*. IEEE, 2012, pp. 165–172.

[7] Y.-J. Zheng, "Water wave optimization: a new nature-inspired metaheuristic," *Computers & Operations Research*, vol. 55, pp. 1–11, 2015.

TABLE I: Research Results for Egg holder's function (1)

Wolf Search Algorithm

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(497,350286104414;413,316038741412)	-445,109840056073
1000	(470,030297278441;485,562031085399)	-747,988554493845
10000	(495,606099411662;401,486352105386)	-787,798958711978

Wolf Search Algorithm with modification

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(482,879289436564;394,047534639969)	-532,10668463655
1000	(362,447760432236;495,061778684641)	-670,883977492652
1000	(459,050439605047;407,631429847158)	-827,9315354643

Water Wave Algorithm

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(416,564008862043;457,568839219198)	-481,75325119652
1000	(482,862816510192;394,327293482761)	-522,678854547558
10000	(429,694043635248;441,956854035126)	-893,969574033705

Water Wave Algorithm with modification

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(386,448251729109;445,815733413126)	-78,0632300803166
1000	(463,005703484177;494,428448618589)	-481,70585333327
1000	(514,777470228624;396,847095860563)	-981,843214134724

TABLE II: Research Results for Easom's function (2)

Wolf Search Algorithm

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(2,34615738054093;3,25216948206172)	-0,365027440014811
1000	(3,9124181950057;3,62785225670219)	-0,276365363830805
10000	(2,96131405698197;2,17447623478923)	-0,212171936285033

Wolf Search Algorithm with modification

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(2,51148351398831;3,47038642199262)	-0,461424318331189
1000	(3,2396818237564;1,77784279537287)	-0,0315487721933061
10000	(2,19667368391374;1,37504338630244)	0,00205881016133928

Water Wave Algorithm

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(2,1929323068787;2,67858712499895)	-0,171094307906584
1000	(3,5790521290987;3,95715529143678)	-0,263663206287755
10000	(3,89253193693819;2,23055503807522)	-0,111170990040294

Water Wave Algorithm with modification

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(3,21630300125866;3,14909067244692)	-0,991576295055776
1000	(2,63754989143347;2,82413371457911)	-0,583388074053213
10000	(1,17472107344061;1,67250233454281)	9,45257018956421E-05

TABLE III: Research Results for Himmelblau's function (3)

Wolf Search Algorithm

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(0,682362732329575;-0,67352732442018)	160,003686302222
1000	(0,633187983945565;-1,88113112742134)	163,753970449623
10000	(-1.0879799313;-0.9926871380)	175.685

Wolf Search Algorithm with modification

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(-0,24526995711274;-0,385508137934612)	178,626060371431
1000	(-0,0459746942138182;-0,795370650848081)	180,212099494084
10000	(-0,234918390044439;-0,621006036932117)	180,680676146925

Water Wave Algorithm

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(-0,526868914965013;-0,343995140559969)	177,35171333301
1000	(-0,086203324182985;-0,733375833711296)	180,378842675413
10000	(-0,594777831153375;-0,345780614924515)	176,703320832286

Water Wave Algorithm with modification

iterations	obtained optimum $\overline{x}$	$\Theta(\overline{x})$
100	(-0,877628832998513;-0,409932400290823)	172,640906231244
1000	(-0,89729838347868;-0,823287643875595)	173,520600953295
10000	(-0,282641141341366;-0,948922069253829)	181,606341140222