Particle swarm method for solving the global optimization problem using the equilibrium coefficient

D T Muhamediyeva

Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Amir Temur street, 108, 100200, Tashkent, Uzbekistan

Abstract. the paper deals with the case when particles are evenly distributed in ravines, using the equilibrium coefficient to update the particle velocity. the method of optimizing a swarm of particles in the case of a "ravine" is one of the most effective approaches for multi-extremal optimization. however, in the existing methods of optimizing the swarm of particles with the "ravine" methods, the number of particles around the ravine is very different from each other, which makes it difficult to find high-quality algorithms in all the ravines. thus, the computational resources are distributed in the ravines in a more balanced way.

1.Introduction

Multi-extremal optimization, which requires finding all optimal (global or local) solutions in the search space, is an urgent task and has a number of real-world applications, such as solving clustering problems [1], wave propagation [2], imitations [3] and design [4]. Evolutionary algorithms (EA) are heuristic search algorithms based on population that were effective in solving complex optimization problems [5]. Because EAs contain a set of solutions, they have natural advantages in finding several optimal solutions in a single pass.

The ravine method is one of the most effective ways to determine optimal solutions for multi-extremal optimization problems. The main idea of the ravine is to maintain a high level of populations using such methods as fitness sharing [6], clustering [7], crowding [8] and limited tournament selection [9]. Although the ravine method proved to be effective for improving the ability of EA to find more optimal solutions with high accuracy, this often requires much more computational effort, since the ravine method usually slows down the speed of the algorithms.

To combat the "ravines" was proposed a number of special techniques. One of them is as follows. From two close points make a gradient descent to the bottom of the "ravine". Then connect the found points to a line and make a large step along it. From the point found, they again descend to the bottom of the "ravine" and take the second ravine step. As a result, moving rather quickly along the "ravine", we approach the desired lowest value of the objective function (Fig. 1). Such a method is rather effective for functions of two variables, however, with a larger number of variables, difficulties may arise.

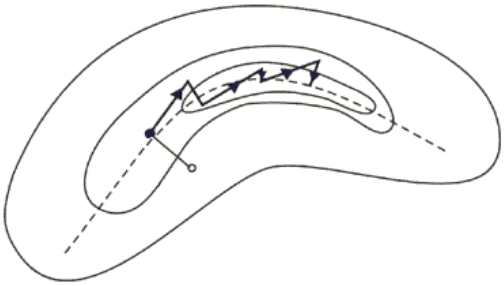


Figure 1. Search for the smallest value of the function in the case of the "ravine"

In fig. 2 shows the function level lines with two local minima at the points O_1 and O_2. Such functions are called multi-extremal.

If, without having before your eyes Fig.2 and not knowing about the multi-extremes of the function, start the search for the smallest value using the method of gradient descent from the point A_1, then the search will lead us to the point O_1, which can be mistaken for the desired answer. On the other hand, if you start the search from the point A_2, then we will be on the right path and quickly come to the point O_2.

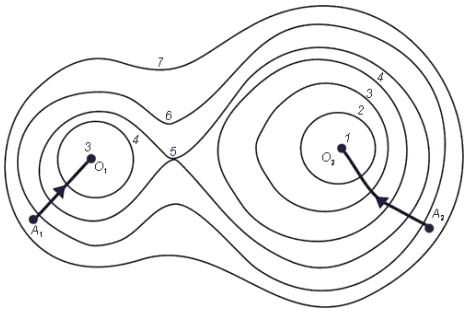


Figure 2. Example of a function with two local minimum at points O_1 and O_2

Particle Swarm Optimization (PSO) is a relatively new version of EA that uses people's experience and public information to find global optimization. It has been proven that PSO is efficient and reliable when facing complex optimization tasks [10].

In 1995, Kennedy and Eberhart [11] proposed for the first time an optimization of a swarm of particles. Although PSO is a global search algorithm based on a population that is similar to EA, it does not have evolutionary operators such as crossover and mutation. In PSO, each solution is represented by a particle that searches for multidimensional space, studying its own experience and past experience of other particles. Its position and speed are updated according to the following expressions:

$$V_i^d(t+1) = w * V_i^d(t) + c_1 * random1_i^d * (pbest_i^d(t) - X_i^d(t)) + \\ + c_2 * random2_i^d * (gbest^d(t) - X_i^d(t)),$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1),$$

where t represents the current generation, refers to the size of the individual i – particle index. c_1 and c_2 – acceleration constants. $random1_i^d$ and $random2_i^d$ – two random numbers from uniformly distributed within the range. Inertia weight w used to balance global and local search performance. $pbest_i^d(t)$ is the best suitability value for the ith particle, whereas $gbest^d(t)$ – the best position found throughout the population.

After a sufficient number of iterations, the particles can unite around the optima, studying the experience of the entire population.

However, the existing PSO in the case of the ravine, faced with the problem of different, non-uniform distribution of the number of particles in different ravines. The imbalance of the particles makes it difficult to accurately identify some of the Optima, which is not grouped with a sufficient number of particles, especially in complex multiextremal optimization task that contains multiple optimal solutions.

To solve the above problem in this paper the equilibrium multiplier to improve the search process. The basic idea is to reduce the differences between the number of particles in different ravines. The equilibrium multiplier encourages the worst particle in the largest ravine to move to a small ravine, which can not only support the accuracy of the solution of the former ravine, but also to increase the probability of finding the best solution in the last ravine. By increasing the number of particles in the small ravine, the algorithm can maintain sufficient diversity in the ravine, which is useful for finding precise solutions. Reducing the number of particles in the largest ravine, the algorithm can drive the particles to explore more search space, which is useful for finding more optimal solutions.

The existing PSO can be divided into two types. The first type focuses on the prevention of the movement of particles in a region of optimum, which were identified in the search process. To achieve this goal it is important to improve the population diversity, in the literature there have been proposed various methods [12]. The second type focuses on the methods of the ravine [13]. The basic idea is to identify multiple Optima using a group of ravines, and then to support the identified Optima until the end of the algorithm.

In the first type of algorithms has been proposed a new PSO, called semiadaptive escape PSO, which uses a special operator-heat, to make the particles more effectively explore the search space [14]. Dynamically generated by a high speed depending on the speed variation that causes particles to simultaneously explore local and global Optima.

In the second type of algorithms are methods of the ravine are used to improve performance. In [15] proposed the method of using a group of subarrays for determining a set of optimal solutions.

2. Formulation of the problem

Let X – compact metric space (optimization set), Φ - bounded below function defined on X (objective function). The task of finding the minimum of the function Φ is to build a sequence of points $x^{(1)}, x^{(2)}, ...,$ из X converging in one of the points $x^* = \arg\min \Phi$ global minimum of function Φ. Convergence types can be different: from convergence in the metric of space X to convergence with some probability.

Assumption of isolation X together with the obligatory assumption about the continuity of the objective function in a neighborhood of the point x^* guarantees that $x^* \in X$, i.e. global maximum Φ in X is achieved.

Thus, let it be necessary to determine such a vector.

$$x^* = (x_1^*, x_2^*, ..., x_n^*)^T, \quad 0 \le x_i \le 1, \quad i \in 1:n,$$

where the objective function $\Phi(x^*)$ accepts the minimum value. We assume that additional restrictions on variable $x_1, x_2, ..., x_n$ taken into account when building the objective function. We write the task in general:

$$\Phi(x) \to \min_{x \in X}, \tag{1}$$

where $x^* = (x_1^*, x_2^*, ..., x_n^*)^T \in [0,1]^n$.

We will consider the following tasks:

Statistical study of the global optimization algorithm and its modernization based on the logistic curve, comparison with the exponential law.

Research results depending on the parameters of the logistic curve.

Identification of the best parameter (set of the best parameters), universal for all classes of functions using decision-making methods.

The entire search is divided into a user-defined number of steps. n_{step}. At each step, according to a certain law, the values of the parameter vector are randomly selected. x^k (k is the step number), and the value of the objective function is calculated $\Phi^k = \Phi(x^k)$. Further, according to the formula

$$\Phi_{\min}^k = \min\left\{\Phi^k, \Phi_{\min}^{k-1}\right\} \quad (2)$$

the smallest value obtained in k steps of the search process is determined. After each calculation using formula (1), the law according to which the values of variables are chosen $x_i (i \in [1:n])$, changed so that the probability of hitting the Δ – the global minimum neighborhood specified by the user, based on the required accuracy of the problem solution, would increase. The information obtained in the previous steps of the search is used for this.

Let us call the interval in which, at the previous steps, the optimal value with a high probability is obtained, the promising interval.

The parameters of the particle swarm, which the user can vary, are: n_{step} – number of steps, $\varepsilon = 1/2\varepsilon$ – accuracy with which the minimum is sought, as well as parameters $p_{\min}$ and $r_{\min}$.

Parameter $r_{\min}$ determines n – dimensional area $s_{\min}$ such that throughout the search process, while $s_k > s_{\min}$, simulation density outside the "promising" area h_k does not change. When s_k getting smaller $s_{\min}$, then h_k begins to tend to zero, i.e. search with increasing intensity is conducted inside I^k.

From value $p_{\min}$ the probability that the minimum in the search process will be inside I^k and height value h_k. Thus, the parameters $p_{\min}$ and $r_{\min}$ in which sense they allow you to set the ratio of "local" and "global" steps to their total number and thereby determine the overall behavior of a swarm of particles.

In the search process, there is an accumulation of information about the nature of the behavior of the function of the target, therefore the width of the “perspective” interval I_i^k logical with each step narrow down from $p_1 = 1$ till $p_{k_m} = p_{\min} < 1$, which entails a decrease in the probability of hitting p_k in I^k. Since it is assumed that the search process converges to a global minimum, it is again logical to assume that

$$\lim_{k\to\infty} p_k = 1, \quad \lim_{k\to\infty} r_k = 0,$$

those it is believed that further starting from , function p_k increases to unity.

3. Particle swarm procedure for solving the global optimization problem.

The particle swarm procedure for solving the global optimization problem consists of four main stages: initialization, building a ravine, updating speed and position, and a local search procedure. At the third stage, the mechanism of the equilibrium coefficient is described.

Stage 1 – Initialization. The first step is random particle generation NP (population size) to form the initial population. All particles are first evenly distributed around the search space. The first step consists of two parts, that is, to initialize the positions of the particles and to initialize their velocities. Both positions and speeds are represented as a vector.

In particular, the first step is performed in accordance with the following expressions:

$$X_i^d = random_i^d (RangL_i^d, RangR_i^d),$$

$$V\max_i^d = RangR^d - RangL^d,$$

$$V_i^d = random_i^d (((-0,5) * \mathrm{Vmax}^d, 0,5 * V\max^d),$$

and $i \in [1, NP]$ – particle index, $d \in [1, D]$ dimension. X_i^d means d – th element i -th particles, V_i^d means d – th element of speed of i -th particles. $random_i^d(a,b)$ returns a random number in the range $[a,b]$. Since if the speed is too high, then the particle can fly out of the search area, the parameter $V\max_i^d$ used to determine the boundary of the d-th element in the particle velocity. $RangR_i^d$ and $RangL_i^d$ represent the upper and lower boundaries of the d-th dimension of the search space, respectively.

A modification of the particle swarm algorithm based on the use of the logistic equation is considered:

$$\frac{dV\max}{dt} = \mu\left(1 - \frac{V\max}{V_\infty}\right)V\max,$$

where $V\max$ – scope of the "promising" area of optimization problem definition, $v = (2q)^n, r$ – its radius, n dimension of the optimization problem, $V_{\lim} = \lim_{t\to\infty} V\max(t) = const$, μ -Malthusian parameter characterizing in our case the rate of change of knowledge about the optimization problem being solved and thus affecting the adaptation of the particle swarm process to solve the global optimization problem. Search is carried out in a single n – dimensional hypercube.

We solve the equation, find:

$$V\max = \frac{1}{\frac{1}{V_\infty} + \left(\frac{1}{V_0} - \frac{1}{V_\infty}\right)e^{-\mu t}},$$

where $V_0 = V\max(0)$. For small values, knowledge increases exponentially, for large values, they approach a certain limit $V_\infty = const$. In our problem, r varies from 0.5 to 0, so we put $V_\infty = 1,\ 0 \le V_0 \le V_\infty$. Then

$$r = \frac{1}{2}\left(1 - \frac{1}{1 + \left(\frac{1}{V_0} - 1\right)e^{-\mu k_{step}/n_{step}}}\right),$$

where k_{step} – step number.

Deduce $V_0 = V_0(\mu, \varepsilon), \mu = \mu(V_0, \varepsilon)$ (under the assumption that the accuracy of the search coincides with the radius of the vicinity of the global minimum). In the last step $2r = \varepsilon$, we get:

$$V_0 = \frac{1}{\frac{\varepsilon}{1-\varepsilon}e^{\mu} + 1},\ \mu = -\ln\left(\frac{\varepsilon}{1-\varepsilon}\frac{V_0}{1-V_0}\right).$$

Thus, the paper considers the exponential law of the change in the "perspective" interval, at which its size at each step decreases in $k' = const$ time $(r = rk')$ and logistic rule with parameter μ .

Consider the search algorithm with extreme parameters of the logistic curve.

Take the radius of a "promising" area r almost unchanged, i.e. $2r = const \approx \varepsilon(\mu \to 0)$. With each step, a uniform throwing of a point occurs within n – dimensional ball of radius - a random choice of direction in which we move. Thus, a descent to a minimum occurs. The probability in this case is higher than in the case of a uniformly distributed search and depends on the type of function of the target.

Take $r: 2r = const \approx \varepsilon(\mu \to 0)$. In this case, there will be practically no narrowing of the gap, and the search will be equivalent to a uniformly distributed search (where are the points at which the

objective function is calculated Φ, are independent implementations of a random element with a uniform distribution on X).

Therefore, it is assumed that the probability depending on the parameter μ increases from the (1) th case, then decreases to the (2) th case, and there is a parameter in which the probability takes its maximum value. We call it the optimal parameter. Next, this parameter is selected.

It was noted that for some functions with different n_{step} There are different optimal parameters.

Stage 2 – Building a ravine.

To solve the multi-extreme problem, the ravine method is used to identify and find several optimums. The best particle in each form is called a descendant. Species are created around descendants and are responsible for drawing closer around optimums. Species will be updated in each generation.

In each generation, particles will be sorted in decreasing order of suitability function. Then all particles are checked in this order to determine if they are suitable for the offspring. One of the important parameters used in this procedure is the radius of the form r, which defines the ravines. The best particle will be selected as a descendant directly and will be included in the descendant set S. S defined as empty in every generation. Next, all descendants will be added to S. Other particles will be checked in order using the radius of the form r. If the distance between the particle and the descendant is less than (or equal to), then the particle is considered a member of the same species with the descendant. If the distance between the particle and the descendant in S more r, then the particle will be considered as a new descendant and placed in S. After sorting all the particles, each particle will belong to the species.

Stage 3 – Updating speed and position.Each particle is controlled by particles $pbest_i$ and $nbest_i$ during each iteration.

The best position detected by particle i is called personal best pbesti, and the current best position of the descendant of its species is called the neighboring best nbesti. These two types of leading particles participate in the velocity equation of the particle swarm method. The equation is similar to the PSO equation, and the only difference is *gbest* replaced by $nbest_i$.

During the search procedure, the number of particles in different gullies is checked gradually. To minimize the difference, an equilibrium factor mechanism (EF) is proposed in which the worst particles of a large ravine will be ejected from the view and move along some other ravines.

The velocity vector (VV) is added to the velocity equation of these several particles so that they move towards the ravine, which has fewer particles. Upon implementation DV, the view will be sorted in descending order of size. The largest and smallest of them will be selected for calculation

$$DV^d = seedS^d - seedL^d,$$

where $seedS^d$ – it is a descendant of the largest species, and $seedL^d$ refers to the descendant of the smallest species.

Meanwhile, all individuals from the largest species will be sorted according to their suitability values. Parameter *DS*, which is defined as the size of the deviation, is calculated to determine the number of particles that will take *EF*. *DS* calculated by the formula:

$$DS = (sizeL + sizeS)/2,$$

where *sizeL* and *sizeS* – the number of particles in the largest and smallest forms, respectively. The equilibrium coefficient mechanism determines *DV* during each iteration and determines which particles will be affected by the vector at the same time. *DV* then used to update particle velocity.

After calculating the VS, the velocities and positions of the particles are updated in accordance with:

$$V_i^d(t+1) = w * V_i^d(t) + c_1 * random1_i^d * (pbest_i^d(t) - X_i^d(t)) +$$
$$+ c_2 * random2_i^d * (nbest^d(t) - X_i^d(t)) + DV,$$
$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1),$$

where expression (3) is for particles that are determined using equilibrium factor (EF). The positions of all particles are updated according to expression (4).

Stage 4. Local search.

In the fourth stage, every *pbest* will be updated by the local search procedure. Each *pbest* will generate a new particle around itself and the best particle will replace another to become new *pbest*. More specifically, for each *pbest* in the population, the operator is finely tuned in accordance with the differences between the original *pbest* with other *pbest*, which is closest to her $pbest_{nearest}$. The local search method, apparently, improves the fine-tuning of the original algorithm, which has proved its effectiveness in many complex problems.

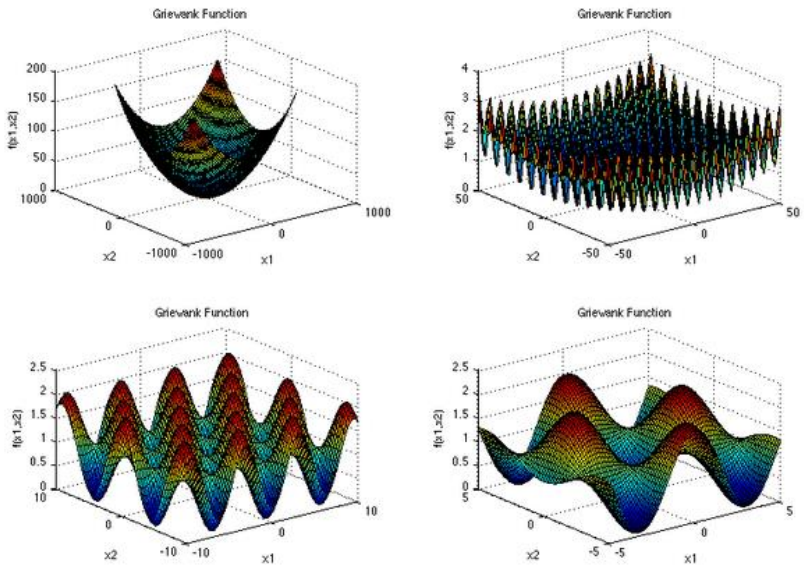
4. Computational experiment.

The proposed algorithm was evaluated on a number of test functions. which are often used to test the effectiveness of new evolutionary algorithms, in addition, complex multi-extremal functions of high dimensionality, practically unsolvable by classical methods, were used [16–19].

All test functions may have a different number of parameters (d). Therefore, it makes sense to run the algorithm to optimize some function first with a small d (for example, 10 or 20), and then with d = 50, 100, 200, ... This will give an opportunity to check the scalability of the algorithm.

Performance of the proposed algorithm is usually estimated by the number of calculations of the objective function. Less is better. The results of the objective function less than 0.001 were also counted as the global minimum found.Because particle swarm algorithms use stochasticity, in order to determine how effective the proposed algorithm is to run it on the same test function several times and only then analyze the result.

1. Griewank function:



$$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

Domain:

$x_i \in [-600,600],\ \mathrm{i}=1,...,d$

Global minimum:

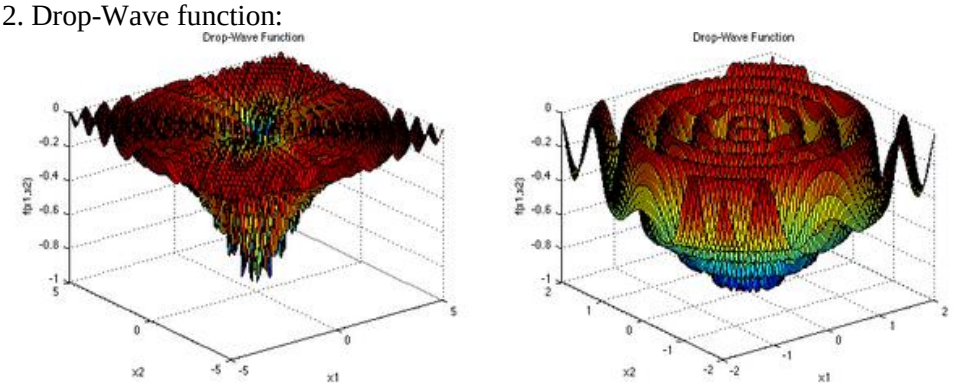
$f(x^*)=0,\ x^*=(0,...,0)$

Particle swarm algorithm*:*

d = 10, the number of finds of the global minimum is 87%, the number of calculations of the objective function is no more than 1200, the maximum value is 0.007251.d = 30, the number of finds of the global minimum is 82%, the number of calculations of the objective function is no more than 1240, the maximum value is 0.074383.

d = 50, the number of finding the global minimum is 75%, the number of calculations of the objective function is no more than 2,700, the maximum value is 0.027492, 40% of the population was selected for crossing.

2. Drop-Wave function:



$$f(x) = -\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2}$$

Domain:

$x_i \in [-5.12, 5.12],\ \mathrm{i} = 1,2$

Global minimum:

$f(x^*) = -1,\ x^* = (0,0)$

Particle swarm algorithm:

d = 2, the number of finding the global minimum is 87%, the number of calculations of the objective function is no more than 1150, the maximum value is –1.003273.

5.Conclusion

This article proposes a way to increase diversity, called the equilibrium factor mechanism. The introduction of the equilibrium factor not only rebalances the distribution of particles in different ravines, but also has the ability to identify a large number of optima in the search space. This helps the algorithms to identify more optima with the necessary accuracy in a single pass and reduce the number of generations. The discussion of the influence of the method of creating parental pairs on the behavior of the proposed algorithm cannot be kept apart from the actual selection mechanism in the formation of a new generation. This method is based on the construction of a new population only of the best individuals of the reproduction group, combining parents and their descendants. The rapid convergence provided by elite selection may, when necessary, be successfully compensated by a suitable method of selecting parental pairs.

References

[1] Naik A, Satapathy S C and Ashour A S Dey N 2016 *Neural Computing and Applications* **1**.

[2] Koper K D, Wysession M E and Wiens D A 1999 *Bulletin of the Seismological Society of America 89* **4** p 978

[3] Goharrizi A Y, Singh R, Gole A M, Filizadeh S, Muller J C and Jayasinghe R P 2015 *IEEE Transactions on Power Delivery 30* **5** p 2128

[4] Mohaymany A S and Gholami A 2010 *Ant colony optimization approach, Journal of Engineering Materials and Technology Transactions of the Asme 120* **1** p 71.
[5] Chu W, Gao X and Sorooshian S 2011 *Information Sciences* **22** p 4909
[6] Goldberg D E and Richardson J 1987 *International Conference on Genetic Algorithms on Genetic Algorithms and Their Application* p 41
[7] Yin X and Germay N 1993 *A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization* p 450
[8] Manner R and Mahfoud S W 1992 *Crowding and preselection revisited, R. manner 390 and B. manderick Parallel* p 27–36.
[9] Stoean C, Preuss M, Stoean R and Dumitrescu D 2010 *IEEE Transactions on Evolutionary Computation*14 **6** p 842
[10] Zhan Z H, Feng X L, Gong Y J and Zhang J 2009 *Eleventh Conference on Congress on Evolutionary Computation* p 1383
[11] Eberhart R and Kennedy J. 1995 *A new optimizer using particle swarm theory*
[12] Shelokar P S, Siarry P, Jayaraman V K and Kulkarni B D 2007 *Applied Mathematics and Computation 188* **1** p 129
[13] Shir O M and Emmerich M 2010 *Evolutionary computation 18* **1** p 97
[14] Li X 2010 *Developing Niching Algorithms in Particle Swarm Optimization*
[15] Brits R, Engelbrecht A P and Bergh F V D 2002 *A niching particle swarm 450 optimizer, in: Conference on Simulated Evolution and Learning* p 692
[16] Muhamediyeva D K 2019 *IOP Conf. Journal of Physics*: *Conf. Series 1210*
[17] Muhamediyeva D T 2019 *IOP Conf. Series: Journal of Physics: Conf. Series 1210*
[18] Muhamediyeva D T 2019 *International Journal of Mechanical and Production Engineering Research and Development* **9** *Issue 2* p 649
[19] Muhamediyeva D K 2019 *International Journal of Mechanical and Production Engineering Research and Development* **9** *Issue 3* p 1095