ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2022. V. LXXV, N3

UDC 621.3.049.77 **MICROELECTRONICS**

DOI: 10.53297/0002306X-2022.v75.3-431

A.V. VARDUMYAN

AN EFFICIENT PRIMARY POPULATION INITIALIZATION METHOD FOR METAHEURISTIC ALGORITHMS

It is widely recognized that convergence capabilities of metaheuristic optimization algorithms can be enhanced by properly chosen initial population. Most of the state-of-the-art initialization techniques are suffering from numerous shortcomings, that ultimately make them non-viable or not efficient enough. To overcome this, this paper proposes a new algorithm for population initialization, which adopts the approach of dividing the search space into nested cubes and picking edge-points for sampling. Based on the data obtained after testing the method on 8 complex benchmark functions against other popular initialization strategies in the scope of WOA algorithm, the proposed approach outperformed all of the candidates in finding the global optimum.

Keywords: metaheuristic algorithm, population initialization, swarm intelligence, evolutionary algorithm.

Introduction. Solving optimization problems generally indicates traversing through the set of all feasible solutions, that satisfy the given constraints in order to identify the one that, depending on the problem statement, either minimizes or maximizes the figure of merit (FOM). These problems, when expressed via mathematical expressions or functions, can have a variable number of properties, control parameters and diverse behavior (continuous/discrete, linear/nonlinear, etc.). So, the general problem formulation is as follows:

$$minimize/maximize\ f(x), subject\ to: x_{j_{lb}} \le x_j \le x_{j_{ub}}\ for\ all\ j = 1,2,\dots,D,$$

where $f(x)$ is the objective function, the value of which for the given $x = [x_1, x_2, \dots, x_D]$ represents the FOM and $x_{j_{lb}}/x_{j_{ub}}$ represent the lower and upper bounds of the search space, respectively.

So far numerous approaches have been devised to tackle this kind of problems, each having their advantages and drawbacks. For instance, among the most popular precise methods being the Gradient Descent, which requires the target function to be convex and needs to calculate its derivatives for each domain point (i.e., it should be differentiable), which for most non-conventional situations is not implementable. On the contrary, there are sets of approximate solution finding approaches that do not impose specific limitations on the objective function

but sacrifice some of the accuracy. These include evolutionary algorithms [Differential Evolution (DE), Genetic Algorithms (GA), etc.] [1] and nature-inspired metaheuristics or so-called swarm-based intelligence [Whale Optimization Algorithm (WOA), Ant-Lion Optimizer (ALO), Gray Wolf Optimizer (GWO), etc.] [2]. Recent particular interest in this type of optimization routines stems from their generalization possibilities, relative ease of implementation and sufficient reduction in the amount of calculations required through iterations. So, it comes as no surprise that more and more natural phenomenon are getting mathematically modelled to simulate certain behaviors and thus designing new algorithms.

One particular step that is crucial for efficient operation in all of them is primary population initialization [3]. Since all of these stochastic techniques operate on populations of individuals (potential solutions or x vectors), properly chosen initial population can give a kickstart to the search process or cause it to stagnate and eventually get stuck in local optima. To surmount that obstruction a number of initialization techniques have been proposed, the primary goal of which is to distribute individuals in the search space as uniformly as possible, thus increasing the odds of finding global optimum [4]. Some of the most widely utilized ones are PRNG, CI, OBL and LHS.

Pseudo-random number generator (PRNG) is by far the most commonly used initialization technique, which generates truly random number sequences scattered across the search space. The only external parameter that can control the generation process is the "seed", that controls the process. The primary downside of this approach is inability to guarantee sufficient coverage of the search space.

Chaotic initialization (CI) is characterized by the chaos theory [5] and has been successfully implemented for metaheuristic algorithms [6]. To generate numbers this approach uses chaotic maps (e.g., Gauss map, tent map, etc.), which produce real values that can be exploited to generate numbers of interest via scaling. The drawback of this technique is the likeliness to miss out on potentially significant features when scaling up to higher dimensions, which is also known as the "curse of dimensionality".

Latin hypercube sampling (LHS) is a statistical method, that generates near-random numbers satisfying additional constraint apart from search space limitations [7]. That being if the area is divided into equally spaced grid mesh, called Latin squares, then each row and each column should have only a single sampling point, which leads adequate representation of problem dimensions in the population [8]. Similar to CI, this strategy also suffers from the "curse of dimensionality".

Opposition-based learning (OBL) is comprised of 2 steps, in which during the first one a population P is initialized for a given number of individuals using

any of the above or other methods based on that the opposite population Y is generated via the following formula:

$$Y[i,j] = UB[j] + LB[j] - P[i,j];\ for\ i = 1, \dots, N; j = 1, \dots, D,$$

where $P[i,j]$ is the j_{th} parameter of the i_{th} individual in population P, $UB[j]$ and $LB[j]$ are the upper and lower bounds for the j_{th} parameter respectively, and $Y[i,j]$ is the corresponding opposite parameter. Finally, a subset of best individuals based on fitness is chosen from both P and Y, which gets returned as the final result [9]. Though it was successfully implemented for metaheuristics [10], the major shortcoming of this method is the need for $2 * N$ function evaluations to generate a population of N individuals, which tends to be computationally expensive if the FOM is not a regular function, but for instance a circuit that needs to be simulated to get the fitness value.

The proposed approach. To overcome the above-mentioned issues from which other algorithms are suffering, a Double-diagonal uniform initialization (DDUI) technique is proposed in which the entire process is divided into several steps. As a first step, the entire search space is divided into $N/4$ nested cubes, the final points will be the 4 edge points of each cube. That is achieved via diagonals that are going through the space and crossing each other (in this paper they are regarded as the main and the secondary diagonals). Examples of what the endpoints might look like for 2D, and 3D planes are presented in Fig. 1 (a) and (b) respectively, in which red marks indicate the final sampling points that should be returned.

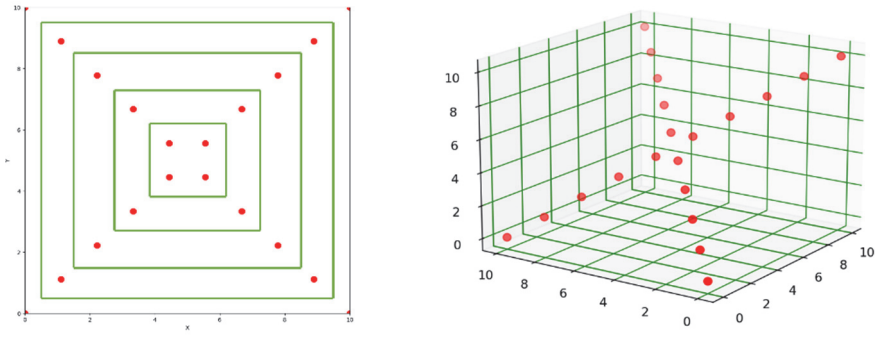

Fig. 1. DDUI sampling points for 2D plane (a) and 3D plane (b)

```
Input: Population size N, values upper bounds UB, values lower bounds LB
number of dimensions D
Output: Initialized population
// Calculate number of points on each diagonal
main_diag, second_diag = 0
if N modulo 4 = 0
    main_diag = second_diag = [N/2]
else if (N − 1) modulo 4 = 0
    main_diag = second_diag = [N/2] + 1
else
    return Error
end if
Allocate memory for population P(x_{i,j})_{N×D}
        or P(x_{i,j})_{(N+1)×D} if main_diag modulo 2 = 1
Initialize main diagonal
Initialize secondary diagonal
// Remove duplicate point
Delete P[main_diag/2] if main_diag modulo 2 = 1
return P(x_{i,j})
```

Algorithm 1. DDUI main routine

```
for i in range(1,2, … , main_diag)
    if i = 1
        P[i] = LB
    else if i = main_diag
        P[i] = UB
    else
        for j in range(1,2, … , D)
            dx = (UB[j] − LB[j])/(main_diag − 1) // Step
            P[i, j] = LB[j] + i ∗ dx
        end for
    end if
end for
```

Algorithm 2. DDUI main diagonal initialization

```
for i in range(1,2, … , second_diag)
    if i = 1
        P[main_diag + i]
            = merge(every second element from LB starting from 1
            + every second element from UB starting from 2)
    else if i = second_diag
        P[main_diag + i]
            = merge(every second element from UB starting from 1
            + every second element from LB starting from 2)
    else
        for j in range(1,2, … , D)
            dx = (UB[j] − LB[j])/(second_diag − 1) // Step
            if j modulo 2 = 0
                new_value = LB[j] + i ∗ dx
            else
                new_value = UB[j] − i ∗ dx
            end if
            P[main_diag + i, j] = new_value
        end for
    end if
end for
```

Algorithm 3. DDUI secondary diagonal initialization

The point coordinate calculation process starts from initial memory allocation and parameter setting as depicted in Algorithm 1. Afterwards the computational process begins for the main and the secondary diagonals, the pseudo-codes for which are presented in Algorithm 2 and Algorithm 3, respectively. In terms of implementation complexity, the entire process involves only several loops and does not conduct any auxiliary calculations and as experiments will further show, the performance does not decline with increasing the dimensionality of the problem.

The only limitation that gets imposed to the user by this technique is that the number of the sampling points cannot be arbitrary, but rather should be either $4 * i$ or $4 * i + 1$, where $i = 1,2,3,$ ..., but this still covers most commonly used round numbers like 20, 40, 100, etc.

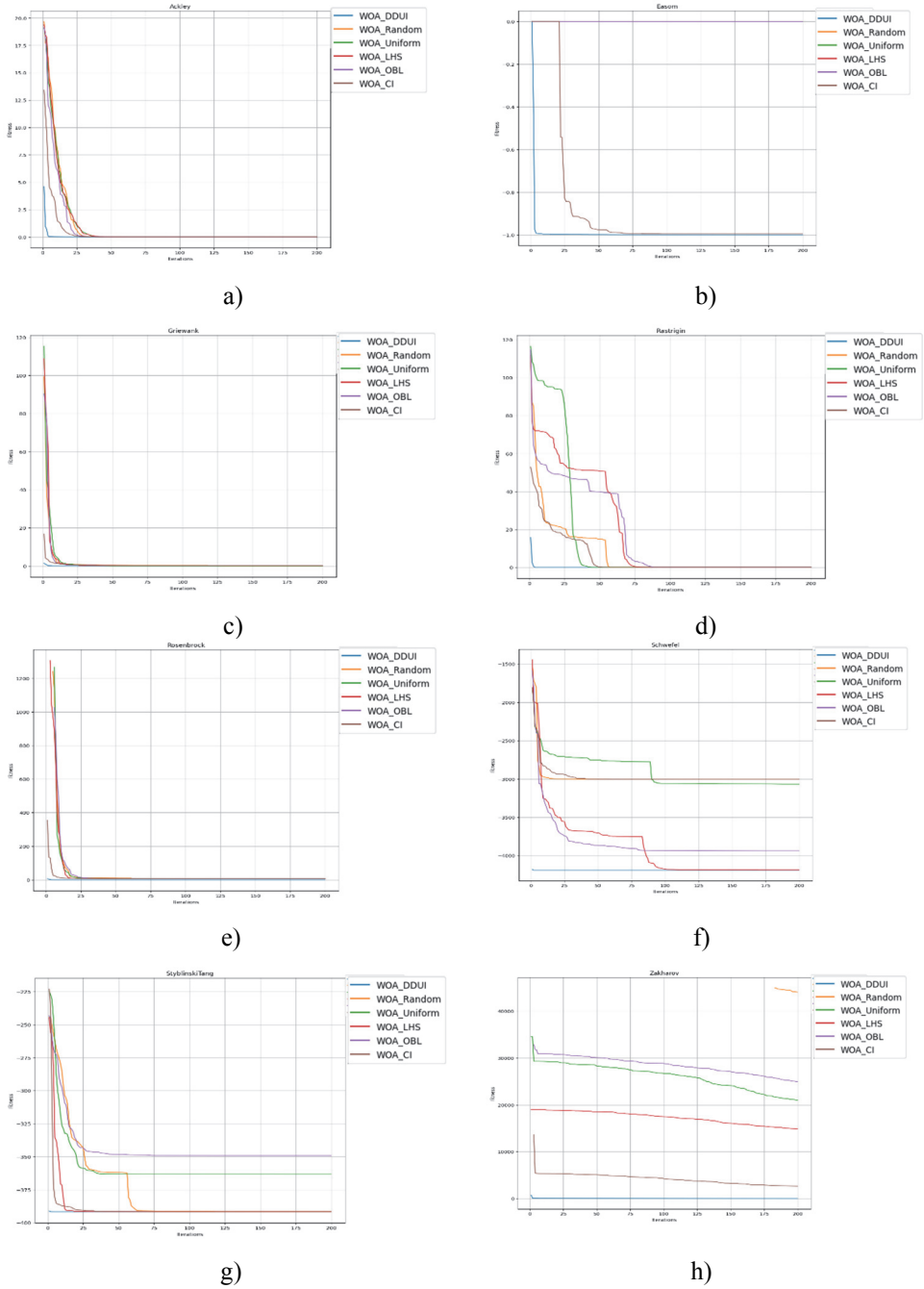

Fig. 2. Convergence plots of different initialization methods on 8 benchmark functions: Ackley (a), Easom (b), Griewank (c), Rastrigin (d), Rosenbrock (e), Schwefel (f), Styblinski-Tang (g), Zakharov (h)

Table 1

Comparison of different initialization methods (dimensions = 10)

Func	Value	DDUI	Uniform	LHS	OBL	CI
1	2	3	4	5	6	7
f_1	Best	0	0	0	0	3.55E-15
	Worst	1.07E-14	1.42E-14	1.42E-14	1.42E-14	1.42E-14
	Mean	5.51E-15	5.86E-15	6.22E-15	6.39E-15	6.04E-15
f_2	Best	-1	0	0	0	-1
	Worst	-0.995393	0	0	0	0
	Mean	-0.998617	0	0	0	-0.42688
f_3	Best	0	0	0	0	0
	Worst	0	0.474477	0.447371	0.248930	0.74061
	Mean	0	0.115378	0.061806	0.055661	0.11250
f_4	Best	0	0	0	0	0
	Worst	1.42E-14	0	38.40678	59.52248	6.20447
	Mean	7.11E-16	0	3.165975	6.439556	0.31022
f_5	Best	5.29E-09	5.156798	0.006609	0.006203	0.00043
	Worst	0.007103	7.165505	7.180006	7.206342	7.19182
	Mean	0.000926	6.488363	6.474888	6.155187	5.64577
f_6	Best	-4189.828	-4189.748	-4189.828	-4189.818	-4189.82
	Worst	-4189.598	-2650.022	-2662.835	-2664.699	-2345.75
	Mean	-4189.793	-3411.158	-3531.479	-3597.201	-3798.93
f_7	Best	-391.6616	-391.6587	-391.6614	-391.6557	-391.659
	Worst	-391.6549	-320.6448	-334.3338	-333.9817	-391.619
	Mean	-391.6604	-370.3126	-383.0364	-378.7673	-391.647
f_8	Best	4.499357	10103.10	8090.126	9829.655	45.7919
	Worst	8.584559	35568.96	42038.54	34923.42	22127.7
	Mean	6.389910	19309.57	23440.08	22512.79	5920.59

Experimental results. To comparatively evaluate the effectiveness of the proposed method a set of 8 benchmark functions was chosen, that contains both unimodal and multimodal functions with narrow global and numerous local optimums. Namely the benchmarks are Ackley (f_1), Easom (f_2), Griewank (f_3), Rastrigin (f_4), Rosenbrock (f_5), Schwefel (f_6), Styblinski-Tang (f_7) and Zakharov (f_8) [11]. As an optimization routine one of the most recent additions to swarm-based intelligence algorithms family, namely the Whale optimization algorithm (WOA) [12] was chosen, that has since gained a lot of popularity and modification proposals [13, 14]. The WOA mathematically models the hunting behavior of humpback whales and offers several routines for exploration and exploitation. As reference initialization candidates for DDUI evaluation Uniform PRNG, LHS, OBL based on uniform PRNG and CI with tent-map as a chaotic map were chosen.

Table 2

Comparison of different initialization methods (dimensions = 50)

Func	Value	DDUI	Uniform	LHS	OBL	CI
1	2	3	4	5	6	7
f_1	Best	3.55E-15	0	0	3.55E-15	3.55E-15
	Worst	7.11E-15	1.42E-14	1.42E-14	1.42E-14	1.42E-14
	Mean	5.15E-15	6.04E-15	6.39E-15	8.70E-15	7.46E-15
f_2	Best	-1	0	0	0	-1
	Worst	-0.215439	0	0	0	0
	Mean	-0.855592	0	0	0	-0.10789
f_3	Best	0	0	0	0	0
	Worst	0	0.312606	0.224252	0	0.25278
	Mean	0	0.015631	0.011213	0	0.01264
f_4	Best	0	0	0	0	0
	Worst	0	5.68E-14	0	5.68E-14	5.68E-14
	Mean	0	2.84E-15	0	2.84E-15	2.84E-15
f_5	Best	1.21E-05	46.94658	47.25961	47.57934	4.90902
	Worst	0.363177	48.48067	48.51746	48.52713	48.5143
	Mean	0.022933	47.94278	47.87552	47.89927	45.7317
f_6	Best	-20949.14	-20945.48	-20947.02	-20947.64	-20948.5
	Worst	-20946.24	-12764.78	-14170.39	-14634.68	-14999.6
	Mean	-20948.70	-17584.73	-18355.21	-18933.11	-18982.1
f_7	Best	-1958.308	-1958.258	-1958.181	-1958.084	-1958.28
	Worst	-1958.237	-1457.922	-1591.349	-1552.201	-1956.58
	Mean	-1958.294	-1915.232	-1886.188	-1834.910	-1957.87
f_8	Best	201.2862	129088.2	149936.4	133597.9	23044.3
	Worst	215.3145	187324.9	223126.8	210587.4	159037
	Mean	208.6043	156493.7	176492.5	168902.9	52760.3

Table 3

Comparison of different initialization methods (dimensions = 100)

Func	Value	DDUI	Uniform	LHS	OBL	CI
1	2	3	4	5	6	7
f_1	Best	3.55E-15	3.55E-15	0	0	3.55E-15
	Worst	1.42E-14	1.42E-14	1.42E-14	1.42E-14	7.11E-15
	Mean	7.11E-15	7.28E-15	5.33E-15	6.04E-15	5.86E-15
f_2	Best	-1	0	0	0	-0.99997
	Worst	-0.003422	0	0	0	0
	Mean	-0.682719	0	0	0	-0.39638
f_3	Best	0	0	0	0	0
	Worst	0	0	0	0	0
	Mean	0	0	0	0	0
f_4	Best	0	0	0	0	0
	Worst	0	2.27E-13	1.14E-13	1.14E-13	0
	Mean	0	1.14E-14	5.68E-15	5.68E-15	0

Continuation of the Table 3

1	2	3	4	5	6	7
f_5	Best	1.45E-07	97.34662	97.80314	97.31989	97.2842
	Worst	2.525692	98.15001	98.12555	98.12313	98.1099
	Mean	0.131511	97.83444	97.96993	97.89779	97.8350
f_6	Best	-41898.28	-41897.52	-41892.49	-41894.94	-41898.2
	Worst	-41895.67	-28071.92	-24079.93	-23955.85	-30007.9
	Mean	-41897.71	-38405.96	-34462.54	-36111.84	-38064.6
f_7	Best	-3916.616	-3916.149	-3916.026	-3916.548	-3916.37
	Worst	-3916.493	-2983.068	-3506.109	-3005.931	-3912.43
	Mean	-3916.594	-3756.602	-3865.889	-3742.011	-3915.02
f_8	Best	408.2037	282207.7	276021.2	277208.6	39730.8
	Worst	428.5733	394584.2	376304.1	394964.3	590361
	Mean	417.9449	331750.8	333909.9	338027.5	150988

For all benchmark problems 10, 50 and 100 dimensions were used for an x vector, the number of individuals in a population was set to 100, while the total number of iterations for finding the optimum was set to 200. On top of that, 20 independent runs were conducted for each of the initialization methods and results are generalized by fetching out “Best”, “Worst” and “Mean” values after the process was over.

Fig. 2 depicts the overall convergence rate of WOA for the chosen initialization methods and DDUI when the problem dimensions equal 10. As can be inferred from the images, without exception for all of the functions DDUI was either finding or getting close to the optimum much faster, consequently for the problems where approximate solution is also acceptable, the entire process can be terminated much sooner.

Tables 1-3 represent the detailed run results data for all the benchmarks and corresponding initialization methods for the problem dimensions of 10, 50 and 100, respectively. It is worth mentioning that the selected number of iterations is generally not sufficient for global optimization problems, where the recommended value is 400…500, but, since in the current problem, we need to evaluate only the initialization method and its impact, instead of performance capabilities of the chosen optimizer, 200 should be sufficient to assess the method efficiency.

Starting from Table 1 (D=10), during those 20 runs DDUI managed to find optimums for 6 out of 8 functions, failing to find one only for Zakharov function and getting very close to Rosenbrock, and, even in that case, the overall performance was incomparably better than for other techniques, which also was generalizing adequately well to state the clear advantage of the method. When D=50 (Table 2), performance does not get diminished, still having some complication with respect

to Zakharov and Rosenbrock, but nonetheless managing to get much closer to optimum than any other technique. It should be stated that for Ackley also the global optimum was not detected, but the impact of optimizer also should not be underestimated, as it involves a lot of randomness when choosing between exploration and exploitation. And finally, Table 3, where D=100, the same issues can be deduced from analyzing the data, but still performing much better than any other counterpart.

Conclusion. The proposed primary population initialization method involves calculation of equally spaced points on the diagonals of the search space, leading to efficient representation of all the search dimensions in the final result. The clear advantages of the method over conventional approaches was discussed and analyzed through test runs on 8 benchmark functions and the results were summarized. The implementation complexity of the approach goes in par with other methods, while imposing a limitation of not being able to choose an arbitrary number for the individuals in the population in other to guarantee a fair distribution.

REFERENCES

1. **Elsayed S.M., Sarker R.A., Essam D.L.** Multi-operator based evolutionary algorithms for solving constrained optimization problems // Comput. Oper. Res. – 2011. – Vol. 38, no. 12. – P. 1877-1896.
2. **Yang X.S.** Nature-Inspired Optimization Algorithms.- Pittsburgh, PA, USA: Academic, 2020. – 310p.
3. **Li Q., Liu S.-Y., Yang X.-S.** Influence of initialization on the performance of Metaheuristic optimizers // Appl. Soft Comput. – Jun. 2020. – Vol. 91, art. no. 106193.
4. **Kazimipour B., Li X., Qin A.K.** A review of population initialization techniques for evolutionary algorithms // IEEE Congr. Evol. Comput. (CEC).-Beijing, China, Jul. 2014. – P. 2585-2592.
5. **Arora S., Sharma M., Anand P.** A novel chaotic interior search algorithm for global optimization and feature selection // Applied Artificial Intelligence. – 2020. – Vol. 34, no. 4. – P. 292-328.
6. **Kuang F., Jin Z., Xu W., Zhang S.** A novel chaotic artificial bee colony algorithm based on Tent map // IEEE Congress on Evolutionary Computation (CEC). – 2014. – P. 235-241.
7. **Helton J.C., Davis F.J.** Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems // Rel. Eng. Syst. Saf. – 2003. – Vol. 81, no. 1. – P. 23-69.
8. **Kotti M., Fakhfakh M., Tlelo-Cuautle E.** Effect of the design space sampling on the design performances // IEEE 9th Latin American Symposium on Circuits & Systems (LASCAS). – 2018. – P. 1-4.

9. **Wang W., Wang H., Sun H., Rahnamayan S.** Using opposition-based learning to enhance differential evolution: A comparative study // IEEE Congress on Evolutionary Computation (CEC). – 2016. – P. 71-77.
10. **Park S., Kim Y., Kim J., Lee J.** Speeded-up cuckoo search using opposition-based learning // 14th International Conference on Control, Automation and Systems (ICCAS 2014). – 2014. – P. 535-539.
11. https://www.sfu.ca/~ssurjano/optimization.html
12. **Mirjalili S., Lewis A.** The whale optimization algorithm // Advances in engineering software. – 2016. – Vol. 95. – P. 51-67.
13. **Ruiye J., Tao C., Songyan W., Ming Y.** A modified Whale Optimization Algorithm based on Chaos Initialization and Regulation Operation // 2019 Chinese Control Conference (CCC). – 2019. – P. 2702-2707.
14. **Feng W., Deng B.** A Modified Multi-Objective Whale Optimization Algorithm with Dynamic Leader Selection Mechanism // 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). – 2020. – P. 985-989.

National Polytechnical University of Armenia. The material is received on 29.08.2022.

Ա.Վ. ՎԱՐԴՈՒՄՅԱՆ

ՄԵՏԱՀԵՎՐԻՍՏԻԿԱԿԱՆ ԱԼԳՈՐԻԹՄՆԵՐԻ ԱՌԱՋՆԱՅԻՆ ՊՈՊՈՒԼՅԱՑԻԱՅԻ ՍԿԶԲՆԱՐԺԵՔԱՎՈՐՄԱՆ ԱՐԴՅՈՒՆԱՎԵՏ ՄԵԹՈԴ

Ընդունված է, որ մետահևրիստիկական օպտիմացիոն ալգորիթմների զուգամիտման կարողությունները կարելի է բարելավել պատշաճ կերպով ընտրված առաջնային պոպուլյացիայով: Ժամանակակից սկզբնարժեքավորման տեխնիկաների մեծ մասն ունի մի շարք թերություններ, որոնք ոչ կենսունակ կամ ոչ արդյունավետ են դարձնում դրանք: Այդ թերությունների հաղթահարման համար առաջարկվում է պոպուլյացիայի սկզբնարժեքավորման նոր ալգորիթմ, որի դեպքում կիրառվում են որոնման տարածքի ներդրված խորանարդների բաժանման մոտեցումը և ծայրամասային կետերի ընտրումը նմուշառման համար: Հիմնվելով առաջարկվող մեթոդի՝ հանրաճանաչ այլ սկզբնարժեքավորման ստրատեգիաների հետ համատեղ 8 բարդ հենանիշ ֆունկցիաներով, Կետ Օպտիմիզացիոն Ալգորիթմի (ԿՕԱ) շրջանակներում թեստավորման արդյունքում ստացված տվյալների վրա, գլոբալ օպտիմումը գտնելու խնդրում առաջարկվող մոտեցումը կատարողականությամբ գերազանցել է բոլոր դիտարկվող մեթոդներին:

Առանցքային բառեր. մետահևրիստիկական ալգորիթմ, պոպուլյացիայի սկզբնարժեքավորում, ամբոխային բանականություն, էվոլյուցիոն ալգորիթմ:

А.В. ВАРДУМЯН

ЭФФЕКТИВНЫЙ МЕТОД ИНИЦИАЛИЗАЦИИ ПЕРВИЧНОЙ ПОПУЛЯЦИИ ДЛЯ МЕТАЭВРИСТИЧЕСКИХ АЛГОРИТМОВ

Принято, что возможности сходимости метаэвристических оптимизационных алгоритмов могут быть улучшены правильно выбранной начальной популяцией. Большинство современных методов инициализации страдают многочисленными недостатками, которые в конечном итоге делают их нежизнеспособными или недостаточно эффективными. С этой целью предлагается новый алгоритм инициализации популяции, в котором используется подход разделения пространства поиска на вложенные кубы и подбор граничных точек для выборки. Основываясь на данных, полученных после тестирования метода на восьми сложных эталонных функциях, в сравнении с другими популярными стратегиями инициализации, в рамках Кит-Оптимизационного Алгоритма (КОА) предложенный подход превзошел всех кандидатов в поиске глобального оптимума.

Ключевые слова: метаэвристический алгоритм, инициализация популяции, роевой интеллект, эволюционный алгоритм.