Hybrid Model of Particle Swarm Optimization and Cellular Learning Automata with New Structure of Neighborhood

Zahra Afsahi
Dept. Electrical and Computer Engineering
Islamic Azad University, Qazvin Branch
Qazvin, Iran
e-mail: afsahi_ai@yahoo.com

Ramin Javadzadeh
Dep. Engineering
Islamic Azad University, Bojnourd Branch
Bojnourd, Iran
e-mail: javadzadeh.r@srbiau.ac.ir

MohammadReza Meybodi
Dept. Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
e-mail: mmeybodi@aut.ac.ir

***Abstract*—In this paper, a population-based optimization algorithm, CLA-PSO, based on new structures of neighborhood is proposed. CLA-PSO is a model obtained by *combining the concepts of particle swarm optimization and* cellular learning automata. Although CLA-PSO produces much better results than the standard PSO, it may trap into local optimum like standard PSO. In this paper two new structures of neighborhood are proposed, which reduce the probability of trapping CLA-PSO into local optimum. The proposed approach is validated by using eight benchmark functions. A comparison is made with the recently published CLA-PSO and standard PSO. The simulation results demonstrate that our approach is highly efficient and competitive specifically in its better general convergence.**

Keywords-particle swarm optimization; cellular learning automata; neighborhood

I. INTRODUCTION

Particle swarm optimization is one of the optimization algorithms which simulates the behavior of birds flocking and school fishing. PSO was first proposed by Kennedy and Eberhart in 1995 [1]. In PSO algorithm all particles move toward the best solution which has been found till now. If this solution was one of the local optima, PSO algorithm converges to local optima finally. Another main drawback of standard PSO is that it cannot control the balance between exploration and exploitation. This problem gets harder in high dimension benchmarks [2]. Many researchers have worked on improving PSO algorithm performance in various ways [1]. For instance, the inertia weight is employed to control the effect of the previous velocities on the current velocity. A large inertia weight facilitates global exploration (searching new areas); while a small one tends to facilitate local exploration. A suitable value of the inertia weight usually provides balance between global and local [14]. Another example of these various ways is social based algorithms including all PSO variations that introduce a new social topology and algorithms that change the way that personal best and neighborhood best positions are calculated [11]. One of the most efficient of these ways is hybrid algorithms, which combine concept from other methods such *as Evolutionary computing [13] genetic algorithm,* Evolutionary programming, Fuzzy systems [12], Ant Colony, Learning automata [9]. In this paper we propose a hybrid method. We combine particle swarm optimization with cellular learning automata. The use of CLA helps PSO to balance between local and global search. Using one LA for each dimension of each particle allows PSO to escape from local optima. In section 2, we give a brief overview of particle swarm optimization algorithm. Section 3 presents an overview of cellular learning automata. Section 4 dedicates all details of our new approach to CLA-PSO with new structures of neighborhood. In section 5, the results which are obtained by our new approach are shown and compared with standard PSO and other kinds of CLA-PSO [5] [6].

II. PARTICLE SWARM OPTIMIZATION

The original PSO was inspired by the social behavior of birds flocking or fish schooling. This algorithm consists of a swarm of particles flying through the search space [1]. Each individual i in the swarm contain parameters for position X_i, velocity V_i, and personal best position y_i, where $X_i \in R^n$, $V_i \in R^n$ and $y_i \in R^n$ while n is the dimension of the search space. The position of each particle represents a potential solution to the optimization. The personal best position associated with a particle i is the best position that particle has visited thus far, *i.e.* a position that yielded the highest fitness value for that particle. If f denotes the objective function, then the personal best of a particle at a time step t is updated as:

$$y_i(t+1) = \begin{cases} X_i(t+1), & f(X_i(t+1)) < f(y_i(t)) \\ y_i(t), & f(X_i(t+1)) \geq f(y_i(t)) \end{cases} \quad (1)$$

Depending on the social network structure of the swarm, and g , best experience of particles exchange

978-1-4244-9252-7/11/$26.00 ©2011 IEEE

among them [4]. For the model, the best particle is determined from the entire swarm.

For the model, a swarm is divided into overlapping neighborhoods of particles. For each neighborhood , a best particle is determined with position . The best particle is referred to as the neighborhood best particle.

In [3] Kennedy and Mendes recommended the Von-Neumann architecture, in which a particle's neighbors are above, below and on each side on a two dimensional lattice, to be the most promising one. For each iteration, in PSO algorithm, the dimension of particle 's velocity vector, and its position vector, is updated as follows:

$$V_i(t+1) = c_1 rand_1(t)\left(y_i(t) - X_i(t)\right) + c_2 rand_2(t)\left(\hat{y}(t) - X_i(t)\right)$$

$$X_i(t+1) = X_i(\quad (2)$$

The PSO algorithm performs repeated applications of the update equations until a specified number of iterations have been exceeded, or until a user-defined stopping criterion has been reached.

III. Cellular Learning Automata

Cellular Learning Automata is a mathematic model for dynamic complex systems which consist of simple components. In such kinds of systems, the behavior of each component is based on its neighbor's behavior and also its own experience. All of these parts have learning capability and act together to produce complex global behavior. Each CLA consists of one CA that equipped by one or more LAs. Learning Automata work as a brain for each cell. It residing in a particular cell determines its action based on its action probability vector. There is a rule that CLA operate under. The rule of the CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in a cell [10]. The basic idea of CLA, which is a subclass of stochastic CA, is to use learning automata to adjust the state transition probability of stochastic CA [6]. The operation of CLA can be considered as follows:

a. The internal state of each cell is specified, based on the action probability vector of its LA. This vector is initiated either based on past experience or randomly.
b. The reinforcement signal of each LA in that cell is determined according to the rule of CLA.
c. Each LA updates its action probability vector based on the reinforcement signal which is determined in second step and the chosen action.

These steps repeated until the desired results are obtained. There is more information about CLA in [7] [8]. In next section we describe our proposed algorithms which are based on new structure of neighborhood.

IV. The Proposed Algorithms

In this section we propose a new CLA–PSO algorithm with new structures of neighborhood. In our proposed algorithm each dimension of each particle is assigned to one cell of CLA. Each cell is equipped with learning automata. LA determines the location of each particle for next iteration. Each learning automata residing in a cell has a set of actions which consist of 2 actions: "following current way" and "following the best experience". The selected action of LA in any iteration specifies the velocity updating method of particle for that iteration. If "following current way" is chosen, determining new velocity will be as (5). If "following the best experience" is chosen in a cell, the velocity for that particle is just determined based on the personal best solution (obtained by that particle) and the global best solution (obtained by the group of particles) as in (3).

One of the main problems in particle swarm optimization algorithm is premature convergence. Whenever the variety among particles, even in one dimension decreases, the movement of all particles on that specific dimension will stop. For this problem, we use the CLA with a global variable. It means that each cell uses both the situation of LA (which are around it) and the global response of environment for making decision about getting reward or punishment.

The global response which is received by all learning automata determines as follows. If the position of a particle improves (better than the previous position) all the LAs which are assigned to all dimensions of that particle will get reward. Otherwise they will get punishment. This response is called βg. Furthermore each LA is received a local response which is called βl. The chosen action is evaluated by determining the variance of neighbor particles' position (V_p). We introduce a new parameter as a threshold of variance (τ_p). Whenever V_p in one neighborhood are less than τ_p shows that on that specific neighborhood, particles are going to be very close to each other, which lead to premature convergence. In this situation LA gets punishment for choosing "following current way" and will get reward if chooses "following the best experience". In contrast a big figure for V_p shows that the variety among particles in a neighborhood is very high. So LA gets reward for choosing "following current way" and gets punishment for choosing "following the best experience". The whole algorithm is as follows:

a. Initialize the position and velocity of the particles. Initialize LA's probability vector.
b. Each dimension of each particle is assigned by a CA and each cell is equipped with one LA.
c. Till the termination condition is met, step *d* to *h* will be repeated.
d. Each LA, based on the probability vector chooses one action.
e. If the chosen action is "following the best experience", the velocity vector will be updated as in (3).

$$V_i^d(t+1) = c_1 rand_1(t)\left(pbest_i(t) - X_i^d(t)\right) + c_2 rand_2(t)\left(gbest_i(t) - X_i^d(t)\right)$$

$$d \in 1, 2, \ldots, N_d \quad (3)$$

f. If the chosen action is "following the best experience", the velocity vector will be updated as in (5).

$$w(t) = w_0 - \frac{t \cdot w_1}{MaxGen} \quad (4)$$

$$V_i^d(t+1) = w(t) * V_i^d(t) + c_1 * rand1_i^d * \left(pbest_i^d(t) - X_i^d(t)\right) + c_2 * rand2_i^d * (gbest^d(t) - X_i^d(t)) \quad (5)$$

g. The global response of is determined as in (6).

$$\beta g_i = \begin{cases} 0 & fitness(X_i(t+1)) < fitness(X_i(t)) \\ 1 & otherwise \end{cases} \quad (6)$$

h. For all cells which are in the same neighborhood, V_p is determined and compare to τ_p as a threshold.

$$\begin{aligned} &if\ variance\left(neighbor\left(X_i^d(t+1)\right)\right) < Tp \\ &\quad if\ LA_i^d.action = 1 \\ &\quad\quad \beta l_i^d = 1 \\ &\quad else \\ &\quad\quad \beta l_i^d = 0 \\ &else \\ &\quad if\ LA_i^d.action = 1 \\ &\quad\quad \beta l_i^d = 0 \\ &\quad else \\ &\quad\quad \beta l_i^d = 1 \\ &end \end{aligned} \quad (7)$$

i. The reward and punishment which is received by LA is determined based on the global and local response of environment:

$$\beta_i^d = \begin{cases} \beta l_i^d & if\ \ \beta g_i = 1 \\ 0 & if\ \ \beta g_i = 0 \end{cases} \quad (8)$$

j. The probability vector will be updated.

A. CLA-PSO with new stuctures of neighborhood

Based on the proposed algorithm, we introduce 2 new kinds of neighborhood. In this structure, the first and last CAs in each row and column are neighbors. It means that the neighborhood is circular. This feature helps the algorithm to maintain the variety during the iteration. In Fig. 1 each cell in C6 neighborhood is neighbor with 6 other cells and in C3 neighborhood is neighbor with 3 other cells. The results show that the first one converges faster than the first one.

B. CLA-PSO with hybrid stucture of neighborhood

In our CLA-PSO each dimension of each particle is assigned with a CA and each CA is equipped with a LA. With this assumption if the number of dimension of population increase the variance will decrease and because of that the variance of particles' velocity will decrease gradually. This decrease leads to premature convergence. In this section, we try to modify the structure of each cell's neighborhood during the iteration. This changing helps the algorithm to increase among population.

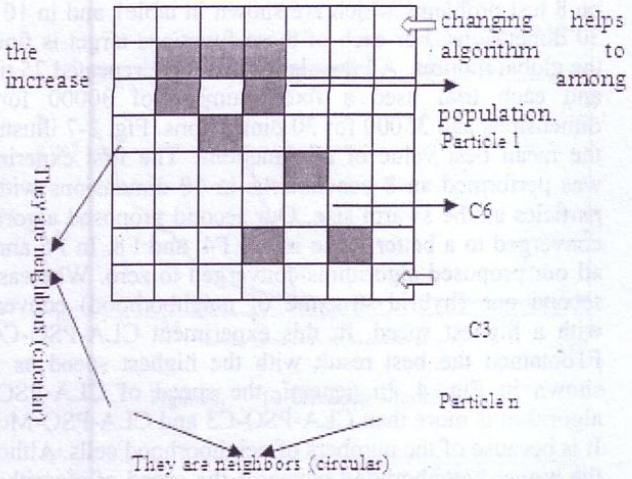

Figure 1. Structure of new neighborhood, structure C6, structure C3, circular.

In each iteration and in step *h* if the variance is less than a threshold the structure of neighborhood will change from C3 to C6 or vice versa. Moreover this hybrid model helps algorithm to explore search space more than before. And increase the variety among particles because C3 and C6 just have one cell in common.

V. EXPERIMENTS

Several simulations were done in order to show the performance of two proposed algorithms.

TABLE I. FUNCTION LIST

Name	Function	
	Formula	*Search range*
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[-100,50]
Rosenbrok	$f_2(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i-1}^2)^2 +$)	[-2.048,2.048]
Ackley	$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$	[-32.768,16]
Greiwank	$f_4(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,200]
Weierstrass	$f_5(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{max}}\left[a^k \cos\left(2\pi b^k(x_i + 0.5)\right)\right.\right.$	[-0.5,0.2]
Rastrigin	$f_6(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 1$	[-5.12,5.12]
Noncontinuesrastrigin	$f_7(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 1$ $y_i = \begin{cases} x_i & \lvert x_i\rvert < 1/2 \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert \geq 1/2 \end{cases}$	[-5.12,2]
Schwefel	$f_8(x) = 418.9829 * D - \sum_{i=1}^{D} x_i \sin$	[-500,500]

The results compare with the results were obtained by standard PSO and CLA PSO (discrete PSO with Moore as a

structure of neighborhood). The simulations were performed on 8 test problems which are shown in table1 and in 10 and 30 dimensions. For each of these functions target is finding the global minima. All the algorithms were repeated 25 times and each trial used a fixed number of 30000 for 10 dimensions and 35000 for 30 dimensions. Fig. 2-7 illustrates the mean best value of all functions. The first experiment was performed an 8 benchmarks in 10 dimensions with 12 particles as the swarm size. Our second proposed algorithm converged to a better value in F1, F4, and F8. In F6 and F7 all our proposed algorithms converged to zero. Whereas the second one (hybrid structure of neighborhood) converged with a highest speed. In this experiment CLA-PSO-C6 in F1obtained the best result with the highest speed as it is shown in Fig. 4. In general, the speed of CLA-PSO-C6 algorithm is more than CLA-PSO-C3 and CLA-PSO-Moore. It is because of the numbers of neighborhood cells. Although the bigger neighborhood increases the speed of algorithm, it leads to premature convergence in high dimension.

The second experiment was performed on 8 functions in 30 dimensions with 12 particles as the swarm size with 35000 iterations. As it is shown in table 2, our second approach algorithm obtained the best result in F2, F4, F5, F7, and F8. CLA-PSO-C3 and CLA-PSO-C6 obtained good result in F1 and F3. In F6 both proposed algorithms converged to zero.

VI. Result

In this paper, a new hybrid algorithm, CLA-PSO with new structure of neighborhood was introduced. In addition in our approach we assigned a CA to each dimension of each particle and each CA was equipped with a LA. These LAs worked as a brain of each cell. Using CLA helps PSO to escape from local optima and make a balance between exploration and exploitation. Experimental results on eight optimization problems show the success and superiority of the proposed algorithm.

References

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," Proceedings of IEEE International Conference on Neural Network, Piscataway, pp. 1942-1948, 1955.

[2] F. V. Bergh, "An analysis of particle swarms optimizers," Survey, Nov, 2001.

[3] J. Kennedy, "Small Worlds and Mega-Minds: Effects of neighborhood topology on particle swarm performance," In proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1931-1938, July 1999.

[4] F. V. Bergh and A. P. Engelbrecht, "Effects of swarm size on cooperative particle swarm optimizers," In proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, USA, pp.892-899, 2001.

[5] M. Sheybani and M. R. Meybodi, "CLAPSO: A new model for optimization," Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Telecommunication Research Center, Tehran, Iran, May 2007.

[6] N. Jafarpour and M.R. Meybodi, "A hybrid method for optimization (Discrete PSO + CLA)," Proceedings of International Conference on Intelligence and Advance Systems (ICIAS2007), Kuala Lumpor, Malaysia, Nov. 25-28, 2007.

[7] M. R. Meybodi, H. Beigy, and M. Taherkhani, "Cellular learning automata and its applications," Journal of Science and Technology, University of Sharif, No. 25, pp. 54-77, Winter 2004.

[8] S. Wolfram, "Cellular Automata," Los Alamos Science, vol. 9, pp. 2-21, Fall 1983.

[9] M. Sheybani and M. R. Meybodi, "PSO-LA: A new model for optimization," Proceedings of 12th Annual CSI Computer Conference of Iran, Shahid Beheshti University, Tehran, Iran, pp. 1162-1169, Feb 2007.

[10] K. S. Narendra and M. A. L Thathachar, "Learning Automata: An Introduction," Prentice-Hall Inc, 1989.

[11] P. Suganthan, "Particle Swann Optimizer with Neighborhood Optimizer," Proceedings of the Congress on Evolutionary Computation, pp. 1962-1958, 1999.

[12] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer," IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

[13] M. Løvberg, T. Rasmussen, and T. Krink, "Hybrid Particle Swarm Optimizer with Breeding and Subpopulation," Proceedings of the Third Genetic and Evolutionary Computation Conference, Vol. 1, pp. 460-476, 2001.

[14] Y. Zheng, L. Ma, L. Zhang, and J. Qian, "Emperical Study of Particle Swarm Optimizer with Incereasig Inertia Weight," In Proceeding of IEEE Congress on Evolutionary Computation, pp. 221-226, 2003.

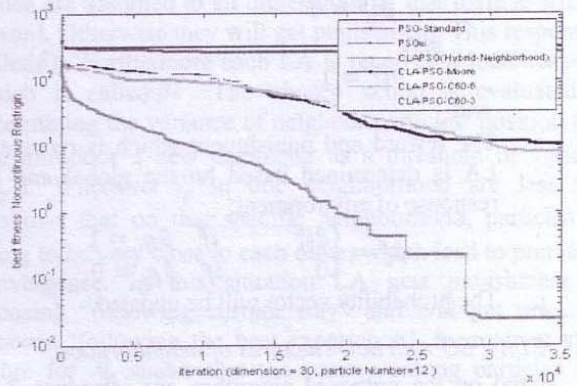

Figure 2. 30-D Noncontinuous Rastrigin's function.

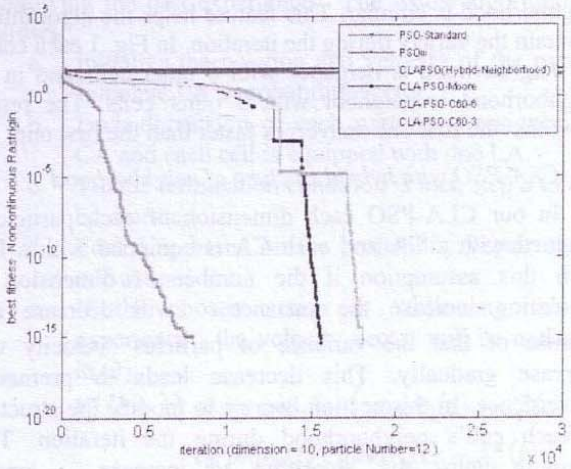

Figure 3. 10-D Noncontinuous Rastrigin's function.

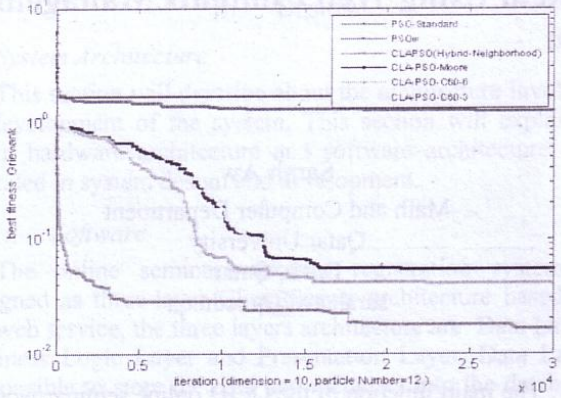

Figure 4. 10-D Griewank's function.

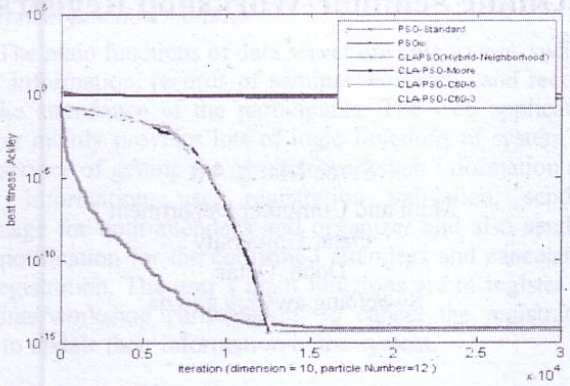

Figure 5. 10-D Ackley's function.

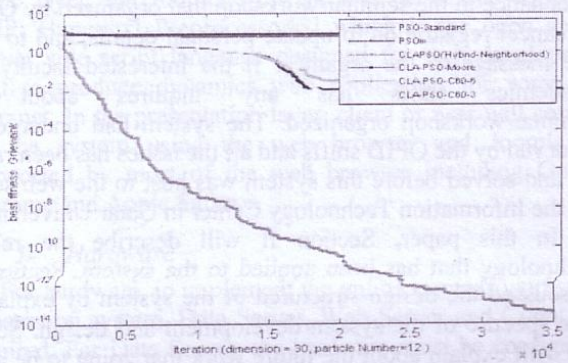

Figure 6. 30-D Griewank's function.

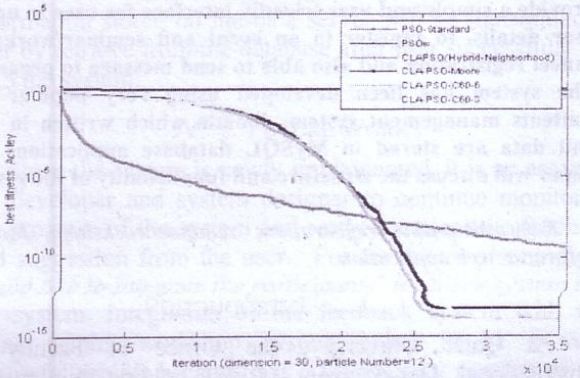

Figure 7. 30-D Ackley's function.

TABLE II. THE RESULTS FOR 30D PROBLEMS.

30D	F1	F2	F3	F4
PSO-Standard	868.09±1812.1	1164.9±5412.3	2.0777±0.0015077	6.767±0.38169
PSOw	753.1±1.6632e+005	1237.4± 3.7665e+005	1.8086±0.11923	9.2272±1.8241
CLA-PSO-Moore	1.644e-058±6.2405e-108	22.184±0.063906	2.8422e-014±1.0691e-027	4.022156e-002± 0.000122287
CLA-PSO-c6	2.187e-065±7.1338e-124	21.953±0.94397	7.1054e-015± 1.0097e-029	7.40218e-002± 0.00050117
CLA-PSO-c3	3.9852e-053±1.869e-097	22.598±0.45261	7.1054e-015±0	3.3307e-002± 0.0003899
CLA-PSO-Hybrid	3.4003e-018±1.2213e-033	21.568±38.446	6.6333e-011±1.8265e-019	4.4409e-013±3.0246e-027
30D	**F5**	**F6**	**F7**	**F8**
PSO-Standard	38.275±1.5736	88.555±2.6325	270.29±119.48	12456±0.094991
PSOw	35.073±1.1368	115.99±1058.8	314.23±707.02	12456±0.086015
CLA-PSO-Moore	2.7626±0.94402	1.0081e-015± 1.0097e-029	23±100.07	12456±0.94533
CLA-PSO-c6	3.5507±15.368	0±0	3.1974± 74.214	12458±0.62061
CLA-PSO-c3	4.7273±55.04	0±0	32.027±137.55	12457±0.3243
CLA-PSO-Hybrid	1.4726e-004±2.6074e-009	1.7764e-015±2.5244e-030	4.8441e-002± 0.0029036	12451±0.00021729