CODMA: A Novel Global Optimization Algorithm based on Open Source Development Model Algorithm with Chaotic Exploration

Hossein Hajipour, Ali Birjandi, Hamed Behzadi Khormuji, Habib Rostami
Computer Eng.Department,Persain Gulf University of Bushehr,Bushehr 75168, Iran
{ho.hajipour, al.birjandi, ha.behzadikhormuji }@mehr.pgu.ac.ir
habib@pgu.ac.ir

Abstract: The open source Development Model Algorithm (ODMA) that was recently introduced has shown its good performance in optimization problems. In this algorithm each point in solution space of the given function has considered as open source software. The considered software (point in the solution space) evolves over time by open source development mechanism. In this paper we present a modified ODMA combine with chaos theory (CODMA). Chaos as a kind of dynamic behavior of nonlinear systems has raised enormous interest in different fields of sciences such as chaos control, pattern recognition, optimization theory and so on. Optimization algorithms based on the chaos theory are stochastic search methodologies that differ from any of the existing evolutionary computation and swarm intelligence methods. General ODMA as like as many evolutionary algorithm may fall into local minimum trap, during the search process, to mitigate this problem we increased the exploration ability of the ODMA, using chaotic behaviour in the Moving toward leading softwares operation. This algorithm is tested by 4 benchmark functions and the results show that the algorithm approach to the global minimum of these functions successfully, also the algorithm is compared with ODMA, GA and PSO and as we shown CODMA have better accuracy and convergence for finding global minima.

Keywords: Optimization, metaheuristics algorithms, chaos theory, open source development algorithm, swarm intelligence, evolutionary algorithm.

1. Introduction

Metaheuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Metaheuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic; or when it is not practical to implement such a method. Many different metaheuristics are in existence and new variants are continually being proposed. These metaheuristic algorithms have been used in wide range of optimization problems such as pattern recognition, machine learning, decision making, industrial planning and etc [1], but there is no specific algorithm to achieve the best solution for all optimization problems and some algorithms may give better solution for some problems than others. Therefore, searching for new metaheuristics algorithm is an open problem.

Different metaheuristic algorithms have been proposed for solving optimization problems [1]. Some of these algorithms were inspired by behavior of biological systems and/or physical systems in nature. For example, Genetic Algorithm is inspired by natural genetic variation and natural selection [3]. Particle Swarm Optimization (PSO) was developed based on swarm behavior of birds and fishes [6], PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle’s position and velocity. Some of other algorithms were inspired by behavior of social phenomena, such as Imperialist Competitive algorithm [2] which is inspired by imperialist competition countries for taking possession of colonies of each other. Another algorithm that were inspired by behavior of social phenomena is Open Source Development Model Algorithm (ODMA) [4]. In ODMA each solution space has considered as open source software and the considered software (point in the solution space) evolves over time by open source development mechanism.

In this paper we propose a new method using chaos theory to move promises softwares toward leading softwares. Many optimization algorithms are inspired from biological systems where order arises from disorder. In these cases disorder often indicates both non-organized patterns and irregular behavior, whereas order is the result of self-organization and evolution and often arises from a disorder condition or from the presence of dissymmetries [7]. Due to these common properties between chaos and optimization algorithms, simultaneous use of these concepts seems to improve the performance. In this proposed method we use some chaotic maps to improve the performance of ODMA and get better solution in functions space. Using the chaos theory the semi-random variation of movement angle causes the proposed algorithm escape from the local optimums during the search process.
To evaluate this new metaheuristic algorithm we examined this algorithm in several benchmark functions that usually tested in metaheuristic algorithms. According to the results we can see that the CODMA is much more efficient in finding global optima and better convergence than ODMA, PSO and GA.

The rest of this paper is organized as follows: section 2 provides a review of related works, section 3 contains a review of ODMA. In section 4 the chaotic maps that generate chaotic sequences for the ODMA are described, in section 5 we present our new metaheuristic algorithm. Experimental results are presented in section 6 and finally in section 7 conclusion are demonstrated.

2. Related Works

Most of metaheuristic algorithm inspired by biological and physical systems and in these recent years’ new optimization algorithms presented that inspired by social phenomena[17]. Some of the most famous of these algorithms are Genetic Algorithm [1], Particle Swarm Optimization [6], Ant Colony Optimization [19], Simulated Annealing [16] and Artificial Immune Systems [22]. During these years, new algorithms are presented by researchers, such as the Imperialist Competitive Algorithm (ICA) [2], Bat Algorithm [18], Gravity Search Algorithm [17], Open Source Development Algorithm (ODMA) [5] and etc.

In some of stochastic algorithms, like SA, the search starts from a single point and continues in a sequential manner[1], however, most of the heuristic algorithms do search in a parallel fashion with multiple initial points, and e.g. swarm based algorithms use a collection of agents similar to a natural flock of birds or fishes[3]. In swarm based algorithm, each point in solution space executes a series of operations and share information with others. Some of well-known and new metaheuristic optimization algorithm will be review in continue[17].

Genetic Algorithms are inspired by Darwin’s theory about evolution. Simply said, the solution to a problem solved by genetic algorithms is evolved. In practice, GA assumes arrays of bits to represent chromosomes [4]. Each iteration improves individuals through simulated evolution; GA tries to find an optimum answer by modeling each solution to a chromosome, whereas ODMA tries to find an optimum answer by modeling each solution as software in open source communities [15].

Particle Swarm optimization is originally attributed to Kennedy, Eberhart and Shi [6] and was first intended for simulating social behavior, as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions, it can therefore also be used on optimization problems that are partially irregular, noisy, change over time, etc.

A new metaheuristic algorithm for optimization that presented in 2007 is an Imperialist Competitive Algorithm (ICA) [2] which is inspired by imperialist competition countries for taking possession of colonies of each other. Another new metaheuristic is a GSA [17]. GSA based on the law of gravity and the notion of mass interactions. The GSA algorithm uses the theory of Newtonian physics and its searcher agents are the collection of masses.

In this paper we propose a new metaheuristic algorithm based on open source development algorithm (ODMA) with combine chaos theory, the operations of ODMA inspired by social phenomena with fast evolution. In ODMA considered each point in the solution space of the given function as an open source software. The considered software (point in the solution space) evolves over time by open source development mechanism. Such as employing features of leading softwares (point which have more optimum solutions) and forking from leading project. Over time, some softwares become leading or show to be promising and some of them can not develop enough. In this paper to have more effective exploration and escape from local optimum, we equip ODMA with chaos techniques to adapt the movement of promises softwares to leading softwares to enhance the escaping capability from a local optima trap.

3. An Overview on Open Source Development Model Algorithm (ODMA)

Open Source Development Model Algorithm (ODMA) is a new metaheuristic optimization algorithm based on open source development model and community [8]. In this algorithm each point in solution space of the given function has considered as open source software. The considered software (point in the solution space) evolves over time by open source development mechanism. Such as employing features of leading softwares (point which have more optimum solutions) and forking from leading project. Over time, some softwares become leading or show to be promising and some of them can not develop enough and will be removed from softwares list (the corresponding points will be deleted in search space and do not process any more).

The algorithm starts with an initial random population called softwares, these softwares initial in $m$ dimensional, in an $m$ dimensional optimization problem, a software is a $1 \ast m$ array. This array has defined as below:

$$
Software = [x_1, x_2, x_3, ..., x_m]
$$

The variable values in the software are represented as floating point numbers. The cost of software is found by evaluating the cost function fitness at the variables $(x_1, x_2, x_3, ..., x_m)$. Then:

$$
fitness(Software) = fitness(x_1, x_2, x_3, ..., x_m)
$$
The algorithm starts with \( n \) initial softwares and \( n_e \) best of them (points with the best solution) chosen as leading software. After initialization the points, to evolve the softwares, these tasks will be run:
- Moving toward leading softwares
- Evolution of the leading softwares
- Forking from leading softwares

In the rest of this section we will describe details of the tasks.

### 3.1 Moving toward leading softwares

In the open source world, softwares evolve to become more popular and receive more resources and users. To do so, open source developers usually analyse exciting leading softwares and use their best practices. To resemble the open source world in our search mechanism, each software (point) selects a leading software based on its ability and position of the leading softwares according to Equation (3), and tries to use its best practices according to Equation (4). Fig. 1 shows the concept of this operation.

\[
softwareLeadingNum = (0, (n_1 - \frac{current\_counter}{\sum_{i=1}^{n_e} leading\_Software[i\_counter]} + 10))
\]  
\( (3) \)

If interval of Random less than 0, then \( softwareLeadingNum = 0 \). To have a smart choice among leading softwares has defined a counter for each software (point).

\[
\begin{align*}
X &= \left( \frac{current\_counter}{leadingSoftwarecounter} + a + b \right) * (d) \\
X_{new} &= \cos(\theta)
\end{align*}
\]  
\( (4) \)

In Equation (4), \( a, b \) is random numbers between \( 0,1 \) and \( d \) is the distance between software and leading software and \( \theta \) is a random number between \( 0, \frac{\pi}{4} \) has defined to search different point around leading software.

![Fig. 1: Moving promise software toward leading software](image)

### 3.2 Evolution of the Leading Softwares

The leading software follows the other patterns to enhance themselves. Software developers release new features’ as alpha version and based on received feedback from their community, the released alpha version may be rolled back or become a new stable version. To model the evolution of the leading softwares has defined a linear equation based on current and past position of leading softwares.

### 3.3 Forking from Leading Softwares

In real world some softwares are introduced based on an exciting leading softwares. In other words, new softwares born by forking from exciting leading softwares. To resemble this phenomena, has considered the following:
- Delete \( n_w \) softwares with least progress in \( i \) th iteration.
- Fork \( n_w \) new software from the leading software .

### 4. An Overview on Chaotic Maps

Chaos as a kind of dynamic behavior of nonlinear systems has raised enormous interest in different fields of sciences such as chaos control, synchronization, pattern recognition, optimization theory and so on [9]. In random-based optimization algorithms, the methods using chaotic variables instead of random variables are called chaotic optimization algorithm (COA).

Optimization algorithms based on the chaos theory are stochastic search methodologies that differ from any of the existing evolutionary computation and swarm intelligence methods. Due to the non-repetition of chaos, it can carry out overall searches at higher speeds than stochastic searches that depend on probabilities [11]. Add the sequences generated from different chaotic map (cm) to some formulas for some operations of ODMA, where it is necessary to make a random-based choice. One-dimensional noninvertible maps are the simplest systems with the capability of generating chaotic motion [12]. In the TABLE 1, we show some of well-known one dimensional maps.
5. Chaotic Open Source Development Model Algorithm (CODMA)

In this paper, we have proposed a new Open Source Development Algorithm using chaos theory. As we say in section 1 the general ODMA may fall into local minimum trap during the search process and it is possible to get far from the global optimum. To mitigate this problem we increased the exploration ability of the ODMA algorithm, using chaotic behaviour in the Moving toward leading softwares. In this way, it is intended to improve the global convergence and prevent being trapped in a local solution.

To using chaotic behaviour in ODMA we extend Equation (4) as follow:

\[
X = CM \times \left( \frac{\text{current.counter}}{\text{leadingSoftware.counter}} + a + b \right) \times (d) \tag{5}
\]

Where \(CM\) in Equation (5) equal to one of the chaotic maps in section 4. What have seen in the software development communities show that promises softwares have chaotic behaviour in inspiring from leading softwares. Usually, among all available features of a leading software, only some of them get attention of developers team of promises softwares. Subsequently a promises software isn’t completely affected by leading software and follow a meaningful different process for its developing, rather than leading software. Aforementioned phenomena embedded in the model by \(CM\) coefficient of Equation (5). \(CM\) coefficient provide desired chaotic behaviour in movement software toward leading softwares and maintain the diversity of population. With \(CM\) specially the type of moving toward leading software have dynamical and non-periodic behaviour that cause increase performance of the algorithm and convergence speed. The CODMA algorithm is summarized in Fig2.

1. Initial software population in random position and evaluate fitness for each Softwares.
2. Compute CM chaotic coefficient for Moving toward Leading Softwares.
4. Moving Leading Software based on their history.
5. Forking from Leading Softwares
6. Check termination condition if false go to step 2 else exit.

Fig. 2 : Pseudocode of CODMA

6. Experimental Results

To validate the performance of our proposed algorithm, we tested it with 4 well-known benchmark functions, these benchmarks are presented in TABLE II. For ease of visualization, we implemented proposed algorithm using java and EvA2 [14]. We compare the accuracy, convergence of CODMA with ODMA, GA and PSO with best selected parameters[15, 5].

<table>
<thead>
<tr>
<th>ID</th>
<th>Function’s Name</th>
<th>Mathematical Represent</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Ackley</td>
<td>(y(x) = -20e^{-0.2} \sum_{i=1}^{n} x_i^2 - e^{\sum_{i=1}^{n} \cos(2\pi x_i)} + 20 + e)</td>
<td>([-32.768, 32.768]), min=0</td>
</tr>
<tr>
<td>F2</td>
<td>Brown3</td>
<td>(y(x) = \sum_{i=1}^{n} x_i^{i^2+1} + (x_i^2 + 1)(i^2+1))</td>
<td>([-1.4]), min=0</td>
</tr>
<tr>
<td>F3</td>
<td>Cigar</td>
<td>(y(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2)</td>
<td>([-10, 10]), min=0</td>
</tr>
<tr>
<td>F4</td>
<td>Zakharov</td>
<td>(y(x) = \sum_{i=1}^{n} x_i^2 + \sum_{i=2}^{n} 0.5(x_i)^2 + \left(\sum_{i=1}^{n} 0.5(x_i)^4\right))</td>
<td>([-5, 10]), min=0</td>
</tr>
</tbody>
</table>
To do so, we compare the proposed algorithm with ODMA, GA and PSO in 1000 iteration and for the purpose of reducing statistical errors, each algorithm is tested 20 times independently for every function and the mean results are used in the comparison. For a fair comparison among algorithms, they are tested using the same population size of 50 and to test this new algorithm in variety dimensions we compare these algorithms in 20, 30 and 50 dimensions. In this comparison CODMA chaotic is Logistic map (TABLE I) and $n_L$ movement (movement of leading softwares based on their history) and $n_s$ of ODMA are 10 and 25, GA mutation and crossover probability are 1 and 0.5 respectively and for PSO cognitive and social parameters are equal 2 respectively and All of these algorithms initialize with 50 of population.

According to results, we can see that the CODMA is much more efficient in finding the global optima and better accuracy than ODMA, GA and PSO in 20, 30 and 50 dimensional in F1 to F4 functions and we can see CODMA outperforms the other algorithm in high dimensional.

The results of the comparison for CODMA, ODMA, PSO and GA are shown in TABLE III to VI.

### TABLE III: Ackley Function

<table>
<thead>
<tr>
<th>Algorithm/Dim</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODMA</td>
<td>9.7045E-09</td>
<td>2.8824E-09</td>
<td>7.6417E-11</td>
</tr>
<tr>
<td>ODMA</td>
<td>9.3487E-05</td>
<td>5.3228E-06</td>
<td>0.000243795</td>
</tr>
<tr>
<td>GA</td>
<td>4.76E+00</td>
<td>6.89E+00</td>
<td>1.19E+01</td>
</tr>
<tr>
<td>PSO</td>
<td>2.78E-07</td>
<td>1.87E-01</td>
<td>1.30E+00</td>
</tr>
</tbody>
</table>

### TABLE IV: Brown3 Function

<table>
<thead>
<tr>
<th>Algorithm/Dim</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODMA</td>
<td>5.7317E-43</td>
<td>8.2296E-32</td>
<td>9.8666E-22</td>
</tr>
<tr>
<td>ODMA</td>
<td>2.1311E-13</td>
<td>8.7788E-11</td>
<td>6.4808E-09</td>
</tr>
<tr>
<td>GA</td>
<td>1.04E+00</td>
<td>7.57E+00</td>
<td>2.76E+00</td>
</tr>
<tr>
<td>PSO</td>
<td>6.87E-05</td>
<td>0.01927061</td>
<td>3.33138427</td>
</tr>
</tbody>
</table>

### TABLE V: Cigar Function

<table>
<thead>
<tr>
<th>Algorithm/Dim</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODMA</td>
<td>9.7045E-09</td>
<td>2.8824E-09</td>
<td>7.6417E-11</td>
</tr>
<tr>
<td>ODMA</td>
<td>9.3487E-05</td>
<td>5.3228E-06</td>
<td>0.000243795</td>
</tr>
<tr>
<td>GA</td>
<td>4.30E-02</td>
<td>3.42E-01</td>
<td>1.90E+00</td>
</tr>
<tr>
<td>PSO</td>
<td>1.46E-05</td>
<td>3.45E-02</td>
<td>4.88E01</td>
</tr>
</tbody>
</table>

### TABLE VI: Zakharov Function

<table>
<thead>
<tr>
<th>Algorithm/Dim</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODMA</td>
<td>8.8102E-33</td>
<td>2.8334E-24</td>
<td>8.6209E-37</td>
</tr>
<tr>
<td>ODMA</td>
<td>1.4320E-15</td>
<td>9.3563E-14</td>
<td>1.9240E-12</td>
</tr>
<tr>
<td>GA</td>
<td>1.35E02</td>
<td>4.08E04</td>
<td>1.40E07</td>
</tr>
<tr>
<td>PSO</td>
<td>1.17E-15</td>
<td>2.73E-14</td>
<td>1.74E-06</td>
</tr>
</tbody>
</table>

### 6.1 Convergence Comparison

For convergence evaluation the result of CODMA, we compare proposed Algorithm with ODMA, PSO and GA in 500 iteration for F2 and F4 functions in 20 and 30 dimensions. The results are shown in figures 1 to 4.

Figure 1 and 2 presents the comparison of convergence for CODMA, ODMA, PSO and GA on Brown3 function in 20 and 30 dimensions. Number of iterations considered for each of these algorithm was 500 time. CODMA proved to be better than general ODMA, PSO and GA and the proposed algorithm has shown a good performance in this function and has been able to escape from the local peaks and reach to global optima.

Figure 3 and 4 shows convergence of the algorithms for Zakharov in 20 and 30 dimensions, which convergence velocity of CODMA on above dimensions works well than the other algorithms. In first 250 iterations of figure 4, PSO works better than CODMA, but CODMA shows better performance after 250 th iteration to the end. it is observed that the quality of global optima solution and the convergence velocity towards the optima point has improved in compare with the other three algorithms.
7. Conclusion

In this paper we presented a modified Open source Development Model Algorithm (ODMA) combine with chaos theory. General ODMA as like as many evolutionary algorithm may fall into local minimum trap, during the search process, to mitigate this problem we increased the exploration ability of the ODMA, using chaotic behaviour in the Moving toward leading softwares. This algorithm is tested by 4 benchmark functions and the results show that the algorithm approach to the global minimum of these functions successfully, also the algorithm is compared with ODMA, GA and PSO and as we shown CODMA had better accuracy and convergence for finding global minima. Therefore our future work will consist in using to solve some of more practical optimization problems and ODMA can extend for discrete and multi objective problem.

References