Computational Research Progress in Applied Science & Engineering

ISSN 2423-4591

CRPASE Vol. 04(01), 22-26, March 2018

Presenting a New Method to Solve Partial Differential Equations using a Group Search Optimizer Method (GSO)

Akram Javadi *, Nasser Mikaeilvand, Hassan Hosseinzdeh

Department of Mathematics, Ardabil branch, Islamic Azad University, Ardabil, Iran

Keywords

Partial differential equations (PDE), GSO algorithm, Dependent error minimization, Graph theory.

Abstract

The analytical methods are not able to solve some of the differential equations. For this reason, the approximate methods like intelligent algorithms are used to reach an answer with the error as low as possible. In this article, a new method has been presented to solve ordinary and partial differential equations (PDEs) which is based on a hierarchical method and the group search optimizer (GSO)-based graph theory. GSO algorithm is an optimization method inspired from the searching behavior of animals and has been implemented based on the producer-scrounger model. It should be noted that according to the aforementioned pattern, the scan mechanisms of animals are symbolically considered here to solve optimization problems. This algorithm has the ability to expand the search space of the genetic algorithm (GA) due to the presence of ranger operator and on the other hand, it has a high convergence speed to find the best answers due to use scrounger model. Therefore, in this article, the best possible answer for PDEs is provided from all possible answers with the least error.

1. Introduction

To solve the differential equations which their analytical answers are not existent or very complicated to calculate, other methods are used to find the closest approximation, resulting in different answers or approximations. One of these methods is the semi-analytic approach for steps and differential transformations. Due to the simplicity of semi-analytic methods to solve delay differential equations (the equations in which the derivative of the function at a certain time is determined based on the function's values at previous times), these methods have gained a great attention in recent decades [1]. Another method is the novel ordered method to solve a system of differential equations that is useful to solve a system of differential equations which include integral functions. In this method, the integral-differential equations are transformed to the matrix equations and in turn, these equations are transformed to a system of linear algebraic equations [2].

Another method is the finite element method which is a numerical technique to find the approximate answer of boundary value problems. This method uses changes account to minimize the error function and produce a robust answer. Like the idea that one can use the relevance of the fine lines connected to each other to approximate a circular environment, the finite elements method also uses the relevance of simpler equational elements on small sub-domains to approximate the equations on a larger domain. Another method used in scientific and applied mathematics calculations to find the numerical answer of differential equations is spectral method. This method uses the fast Fourier transform (FFT), in other words, it writes the answers of the differential equations as a sum total of certain basic functions (for example, Fourier series as a total sinus), then, it chooses the coefficients from the total which have the most compatibility with differential equations. The spectral and finite elements methods are close together and based on the same idea. The main difference between is that the spectral method uses the basic functions that are non-zero on the whole domain while the finite elements method uses the basic functions that are non-zero on small sub-domains (not on the whole domain). In other words, the spectral method chooses a general way to achieve the goal while the finite elements method chooses a local way. In Wagdy [3], a new approach algorithm called DESP has been presented for differential assessment to solve the stochastic programming problems. The proposed algorithm presents a new triangular mutation rule based on the triangle convex combination vector and the difference vector between the best and worst individuals of the three vectors selected randomly. The new proposed method uses the mutation operator to increase the

* Corresponding Author:
E-mail address: akramjavadi97@gmail.com– Tel, (+98) 9143579769

Received: 30 January 2018; Accepted: 10 March 2018

general and local search abilities and the convergence speed compared to conventional DE. DESP uses the constraint management technique of Deb without any additional parameter and based on the feasibility and total constraint violations. Furthermore, a new dynamic has been used for the training and management of equality constraints and two models of stochastic programming (SP) have been considered, multi-objective fractional stochastic programming and linear stochastic programming problems. The comparison among the DESP, conventional DE, partial swarm optimization (PSO) and GA shows that the proposed DESP is comparable with (and in some cases better than) other algorithms in terms of the quality of the answer, efficiency and the robustness of the problems. In the study conducted by Hongliang [4], a new adaptive dynamic programming algorithm has been developed based on the repetition policy to solve multi-player non-zero-total differential game for continuous-time nonlinear systems. This algorithm is mathematically equal to repetition in a Banach space. The implementation has been presented using neural networks so that an elite neural network has been used to learn the value function and an action neural network that share same parameters with the corresponding elite neural network has been used to learn optimal control policy for each player. All the action neural networks are online and real-time and updated continuously. A simulation example has been provided to prove the performance of the proposed method. An unusual and non-traditional method to solve ordinary and partial differential equation has been presented in [5] in which a completely heuristic and novel method has been used to solve the problem. This method is based on the ant colony algorithm, various examples have been presented and the results have been compared with the results of traditional methods. Many methods have been presented to solve ODEs and PDEs. Recently, new methods have also presented to solve ordinary differential equations based on the intelligent algorithms like genetic and ant colony algorithms that have provided results with as low as possible error [6].

In this article, a new method is presented to solve partial differential equations. This method is based on the GSO programming. This approach produces trial answers and try to maximize the value of objective (or fitness) function which is the inverse of the error. In the following, some descriptions are presented about GSO algorithm and then the graph theory method and the hierarchical method are presented to solve differential equations based on the GSO planning method in the next section. Finally, some studies are presented to solve various types of systems including PDE equations and the results are discussed.

2. GSO Algorithm

In fact, the purpose of optimization is to find an optimal point in a search space and this procedure is comparable with resource search procedure of animals. GSO algorithm that takes advantage of population-based logic is an appropriate option for the optimizations of electricity field [7].

In a n-dimension search space, the i^{th} member in k^{th} repetition is called X_i^k $(\in \mathbb{R}^n)$. Head angle is $\varphi_{i1}^k, \dots, \varphi_{i(n-1)}^k) \in \mathbb{R}^n$ and head direction is $D_i^k(\varphi_i^k) = (d_{i1}^k, \dots, d_{in}^k) \in \mathbb{R}^n$ that can be calculated from φ_i^k through converting polar coordinates to Cartesian ones.

$$d_{i1}^k = \prod_{p=1}^{n-1} \cos(\varphi_{ip}^k) \quad (1)$$

$$d_{ij}^k = \sin\left(\varphi_{i(j-1)}^k\right) * \prod_{p=i}^{n-1} \cos\left(\square_{ip}^k\right) \ .(j = 2. \dots . n-1) \quad (2)$$

$$d_{in}^k = \sin\left(\varphi_{i(n-1)}^k\right) \quad (3)$$

As mentioned, in each repetition, a member of the group that is placed on the best point in terms of objective function value is selected as producer. The search field of a producer in a n-dimension space can be determined based on the maximum of viewing angle $\theta_{\max} \in \mathbb{R}^{n-1}$ and the maximum of chase interval $I_{\max} \in \mathbb{R}^1$ (Figure 1).

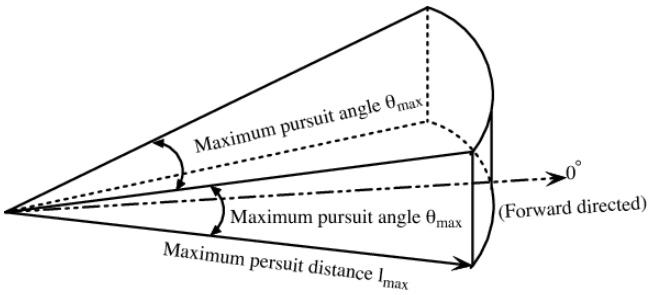

Figure 1. Visibility in a 3-dimension space

The search behavior of producer is modeled as follows. Producer X_p^k produces three new search points randomly to choose three new positions. The first position is 0 degree and its distance from curthe rent position is selected randomly as

$$X_z^k = X_p^k + r_1 * l_{max} D_p^k\left(\varphi^k\right) \quad (4)$$

The second position is randomly selected from the right side of the producer

$$X_r^k = X_p^k + r_1 * l_{max} D_p^k\left(\varphi^k + r_2\theta_{max}/2\right) \quad (5)$$

And the third one is randomly selected from the left side

$$X_r^k = X_p^k + r_1 * l_{max} D_p^k\left(\varphi^k - r_2\theta_{max}/2\right) \quad (6)$$

The p^{th} producer finds the best position with best resources in p^{th} considered function. If the fitness value of the best point is better than current position's one, then this point is transferred to the new position, otherwise $X_p^{k+1} = X_p^k$ remains in current position and produces a new head angle through rotation

$$\varphi_p^{(k+1)} = \varphi_p^k + r_2\varphi_m ax \quad (7)$$

αmax is the rotation angle. If the producer couldn't find a better position after a repition, then it could rotate the head toward the previous angle.

$$\varphi_p^{k+a} = \varphi_p^k \quad (8)$$

where a is a constant value.

Except for producer, 80% of remained members of the group are selected as scroungers. Scroungers keep searching to find food resources around the producer. According to Figure 2, the movement paths of scroungers are very visible. It is clear that scroungers are finally converging to a global optimal point. It should be noted that if a scrounger or ranger finds a more appropriate point (in terms of objective

function) in the search process of GSO then, this new point is replaced with the producer in the next repetition. Now the mentioned pattern for the producer will be repeated in the new point.

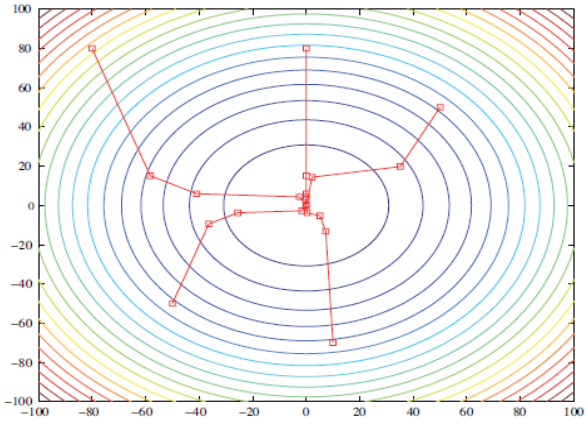

Figure 2. The paths of 5 scroungers that are moving toward producer in 5 repetitions

The regional copy format has been used for scroungers and their movements toward producer. In kth repetition, the regional copy behavior is relationally modeled as

$$X_j^{k+1} = X_j^k + r_3 * \left(X_p^k - X_i^k\right) \quad (9)$$

For this purpose, $r_3 \in \mathbb{R}^n$ is formed from the numbers between 0 and 1.

The rest members of the group -except for producers and scroungers- are called rangers. The behavior of rangers is the first phase of search behavior that is started without direction to a special resource. The search strategy of rangers includes random walking and systematic search that help to find effective resources. In fact, Rangers play the role of mutation in genetic algorithm and direct the algorithm to an absolute optimal answer when it is trapped in local optimums. Again, a random head angle is produced for rangers using Eq. (10). Then, the random distance (interval) is obtained using the following equation.

$$l_i = ar_1 l_{max} \quad (10)$$

The following equation is used to calculate the new positions of rangers

$$X_r^{k+1} = X_r^k + l_r D_r^k\left(\varphi^{(k+1)}\right) \quad (11)$$

As it is obvious in resource search pattern of animals in the real world, after crossing the regions designated for food search, the animal tends to return to her/his designated territory in order to not to lose her/his chance to find food or other desired resources. This pattern is used in GSO algorithm to return the algorithm to the allowed range of variables. It should be noted that the following algorithm is used to find the maximum interval (distance).

$$l_{max} = \|U - L\| = \sqrt{\sum_{i=1}^{n}(U_i - U_i)^2} \quad (12)$$

The GSO algorithm as a powerful method in continuous variables field can acceptably converge to general optimal answers (or near them) and its speed and accuracy are comparable with similar optimization methods with the initial-population-based pattern. Since the variables that are mostly used as the input of objective functions in the electricity field and engineering fields are considered as discrete values, we must modify the proceeding procedure of algorithm using the basics of probabilities and similar basics to reach the optimal answers properly. For this purpose, the discrete model of the algorithm has been implemented in this article for the first time and some acceptable results have been obtained in the field of equipment placement. The flowchart of problem-solving has been presented in Figure 3.

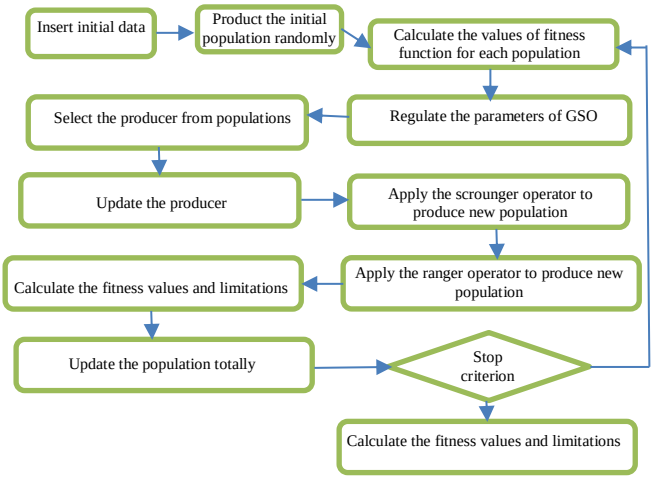

Figure 3. The flowchart of problem-solving by GSO algorithm

3. Evaluating the Fitness of PDEs

Here, we only consider the two and three-variable elliptic PDEs with Dirichlet boundary conditions. The extension of this process to other types of boundary conditions and higher dimensions is straightforward. PDE is described as

$$f\left(x, y, \psi(x,y), \frac{\partial}{\partial x}\psi(x,y), \frac{\partial}{\partial y}\psi(x,y), \frac{\partial^2}{\partial x^2}\psi(x,y), \frac{\partial^2}{\partial y^2}\psi(x,y)\right) = 0 \quad (12)$$

with $x \in [x_0.x_1]$ and $y \in [y_o.y_1]$, the Dirichlet dependent boundary conditions are described as follows

$$E(M_i) = \sum_{j=1}^{N^2} f\left(\begin{matrix} x_j, y_j, M_i(x_j,y_j), \frac{\partial}{\partial x}M_i(x_j,y_j), \frac{\partial}{\partial y}M_i(x_j,y_j), \\ \frac{\partial^2}{\partial x^2}M_i(x_j,y_j), \frac{\partial^2}{\partial y^2}M_i(x_j,y_j) \end{matrix}\right)^2 \quad (13)$$

$$v_i = E(M_i) + P(M_i) \quad (14)$$

In Burgess [8], $P(M_i)$ is not considered and have a zero value. In all of these functions, we consider the final fitness function as follows for more accurate evaluation. Where if the output of error function is zero, the value of fitness function is 1. In other words, the more the error, the more the value of fitness function tends to zero and the more the system get away from the main answer.

$$Fittnes_i = \frac{1}{1+v_i} \quad (15)$$

where v_i is the value of total error function that was mentioned in the previous section. This equation can be used for all kinds of PDEs.

4. Problem-solving Approach

In this approach, we divide the characters into symbol, function, and operator and the populations produced in each level are determined based on the symbols and functions. These symbols, functions, and operators are listed in the Table 1 with their codes. In the hierarchical theory presented in this article, the algorithm works in multi-level mode, the number of different populations variables are different, and the system produces the next sub-levels according to the type of produced population and used functions. In this mode, if a function is created in the second sub-level, the third sub-level is created and this process continues until the function isn't used in the system. Here, it is not considered any limitation for the levels; the hierarchy of the created characters isn't considered; and these levels are determined based on the system's conditions. On the other hand, in this theory, the characters are divided into three categories and the numbers are determined based on the symbol 1 and are in the range of [-20, 20]. These numbers can be integer or decimal. In addition, other characters are placed in the function according to the considered number in its equivalent function. For example, a population is as [18 12 1 14 11], in this case, a population should be created due to the use of 18 that is equivalent to cos; in this population, the sub-system is on the second level [2 14 20] and due to the use of 20 (exp), a sub-system is also created on the third level that is [9] here. In this condition, the equation created by this theory is $\cos(2 * \exp(t)) + 1 * t$ that have three levels.

Table 1. Symbols, functions, and operators of the system

	Symbols	Operators	Functions
Characters	t , Number	^ / * - +	Sin, cos, Log, Exp
The code related to each character	1,11	12,13,14,15,16	17,18,19,20

- **PDEs**

For the following PDE

$$\frac{\partial \psi}{\partial x} + \frac{\partial \psi}{\partial y} = 2(x + y)$$

The proposed answer of the program is as follows

```
Loop Index = 3
Best Fitness = 1
Step's Best Solution = x*(y)+x*(y)
**********
Final Solution = x*(y)+x*(y)
```

which is equal to $2xy$ and is obtained in the third step. The value of error in this equation is zero and the system has found the certain answer.

For the following PDE

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial \psi}{\partial y} = -\psi + \cos(x + y)$$

The proposed answer of the program is as follows

```
Loop Index = 82
Best Fitness = 0.96576
Step's Best Solution = 1.0151^(x)*sin(x+y)
**********
Final Solution = 1.0151^(x)*sin(x+y)
```

which is obtained in step 82. For the considered range of [-0.5, 0.5]

$$1.0151^{x} \times \sin(x + y) \equiv \sin(x + y)$$

For the following PDE,

$$\frac{\partial \psi}{\partial x}\frac{\partial \psi}{\partial y} = \psi$$

After a rapid convergence, the proposed answer is as follows

```
Loop Index = 3
Best Fitness = 0.97551
Step's Best Solution = (-x)*(y)*log(abs(0.049463+0.53576))+(-x)*(y)*log(abs(0.049463+0.53576))
**********
Final Solution = (-x)*(y)*log(abs(0.049463+0.53576))+(-x)*(y)*log(abs(0.049463+0.53576))
```

By a simplification, we have

$$-xy \times log\big(abs(0.049463 + 0.53576)\big) - xy \times \log\big(abs(0.049463 + 0.53576)\big) =$$

$$-2xy \times log\big(abs(0.049463 + 0.53576)\big) = C_1 xy$$

And the validity of the answer is confirmed by a simple check. The results obtained by this algorithm is compared with the results obtained by GA, GSO, and ACO algorithms in the Table (2) and it is considered that the best and fastest answer is related to the GSO algorithm.

Table 2. The results obtained by this algorithm is compared with the results obtained by GA, GSO, and ACO algorithms

Examples	Algorithm	Fitness function value	Repetition number
The first example	GSO algorithm	1	3
	GA algorithm	0.98541	21
	PSO algorithm	1	14
The second example	GSO algorithm	0.96576	82
	GA algorithm	0.81268	74
	PSO algorithm	0.89245	30
The third example	GSO algorithm	0.97551	3
	GA algorithm	0.96980	89
	PSO algorithm	0.94962	8

5. Conclusions

In this article, a new method has been presented to solve partial differential equations based on a graph theory-based method and a hierarchical method that uses a metaheuristic method called Group Search Optimizer (GSO). In addition to this algorithm, the production of populations for different generations is done as multi-level and based on the graph theory to reach an answer with as low as possible error. In fact, the used grammar method has been developed and improved in this article that provides a faster and more accurate answer. The obtained results also showed the accuracy and speed of this method. In the theory presented in the article, the limitations related to the created levels have been removed; in addition, according to the features of the

ranger operator for population production, the numbers can be produced as integer and decimal.

References

[1] J.P. Richard, Time-delay systems: An overview of some recent advances and open problems, Automatica 39 (2003) 1667–1694.

[2] F. Mirzaee, S.F. Hoseini, A new collocation approach for solving systems of high_order integro_differential equation with variable, Applied Mathematics and Computation 311 (2017) 272–282.

[3] A.W. Mohamed, Solving stochastic programming problems using new approach to Differential Evolution algorithm, Egyptian Informatics Journal (2016).

[4] H. Li, D. Liu, D. Wang, Adaptive Dynamic Programming for Solving Non-Zero-Sum Differential Games, 3rd IFAC International Conference on Intelligent Control and Automation Science, Chengdu, China, September 2–4 (2013).

[5] J.D. Lambert., Numerical methods for Ordinary Differential Systems: The initial value problem. John Wiley & Sons: Chichester, England, 1991.

[6] H. Cao, L. Kang, Y. Chen, J. Yu, Evolutionary Modeling of Systems of Ordinary Di_erential Equations with Genetic Programming, Genetic Programming and Evolvable Machines 1 (2010) 309–337.

[7] S. He, Q.H. Wu, J.R. Saunders, Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior, IEEE Transactions on Evolutionary Computation 13 (2009) 973–990.

[8] G. Burgess, Finding Approximate Analytic Solutions to Differential Equations Using Genetic Programming, Surveillance Systems Division, Electronics and Surveillance Research Laboratory, Department of Defense, Australia, (2009).

[9] J. Nieminen, J. Yliluoma., Solving Di_erential Equations with Radial Basis Functions: Multilevel Methods and Smoothing, Advances in Computational Mathematics 11 (2009) 139–159.