INFORMATICA, 2024, Vol. 35, No. 3, 453–481

DOI: https://doi.org/10.15388/24-INFOR563

Solution of Inverse Problem for Diffusion Equation with Fractional Derivatives Using Metaheuristic Optimization Algorithm

Rafał BROCIEK[1,*], Mateusz GOIK[1], Jakub MIARKA[1], Mariusz PLESZCZYŃSKI[1], Christian NAPOLI[2]

[1] *Department of Mathematics Applications and Methods for Artificial Intelligence, Silesian University of Technology, 44-100, Gliwice, Poland*
[2] *Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Via Ariosto 25, 00185 Rome RM, Italy*
e-mail: rafal.brociek@polsl.pl

Received: March 2024; accepted: June 2024

Abstract. The article focuses on the presentation and comparison of selected heuristic algorithms for solving the inverse problem for the anomalous diffusion model. Considered mathematical model consists of time-space fractional diffusion equation with initial boundary conditions. Those kind of models are used in modelling the phenomena of heat flow in porous materials. In the model, Caputo's and Riemann-Liouville's fractional derivatives were used. The inverse problem was based on identifying orders of the derivatives and recreating fractional boundary condition. Taking into consideration the fact that inverse problems of this kind are ill-conditioned, the problem should be considered as hard to solve. Therefore,to solve it, metaheuristic optimization algorithms popular in scientific literature were used and their performance were compared: Group Teaching Optimization Algorithm (GTOA), Equilibrium Optimizer (EO), Grey Wolf Optimizer (GWO), War Strategy Optimizer (WSO), Tuna Swarm Optimization (TSO), Ant Colony Optimization (ACO), Jellyfish Search (JS) and Artificial Bee Colony (ABC). This paper presents computational examples showing effectiveness of considered metaheuristic optimization algorithms in solving inverse problem for anomalous diffusion model.

Key words: metaheuristic algorithms, inverse problem, fractional derivative, time-space fractional diffusion equation, fractional boundary condition, identifying parameters, numerical computation.

1. Introduction

Nowadays, when all kinds of computational and simulation methods are becoming more and more important and the processing power of computers is increasing, it is worth developing and looking for new applications of this type of tools. In practice, there are many classical numerical methods that are successfully used, developed and adapted to current problems. On the other hand, in recent years, scientists have been developing artificial

*Corresponding author.

intelligence methods, which include metaheuristic optimization algorithms. In this paper, we focused on combining a classical numerical method for solving a differential equation and selected heuristic algorithms for solving the inverse problem.

In the scientific literature, many works on the use of fractional derivatives to model many processes occurring in physics and engineering can be found (Bhangale *et al.*, 2023; Brociek *et al.*, 2017, 2019; Sowa *et al.*, 2023; Bohaienko and Gladky, 2023; Koleva *et al.*, 2021). In particular, derivatives of this type are used to model anomalous diffusion. As an example, heat flow in porous materials can be mentioned (Brociek *et al.*, 2017, 2019). In article (Brociek *et al.*, 2019), the authors model the phenomenon of heat flow in a porous medium. For this purpose, several mathematical models were used, including those based on fractional derivatives. The results from numerical experiments were compared with measurement data. Models that used fractional derivatives proved to be more accurate than the model with integer-order derivatives. Sowa *et al.* (2023) used fractional calculus and presented a voltage regulator extended model. The approach presented in the paper for modelling a voltage regulator has been verified with measurement data, and the non-integer order Caputo derivative proved to be an effective tool. Another application of fractional derivatives in mathematical modelling can be found in the paper (Bohaienko and Gladky, 2023), where a model for predicting the dynamics of moisture transport in fractal-structured soils was presented. The model incorporates the Caputo derivative. The numerical solution was obtained using the Crank-Nicholson finite-difference scheme. More examples and trends in the application of fractional calculus in various scientific fields are available in the publications (Hristov, 2023b; Obembe *et al.*, 2017; Ionescu *et al.*, 2017). Publications related to numerical methods in the field of fractional calculus are also worth mentioning (Ciesielski and Grodzki, 2024; Hou *et al.*, 2023; Arul *et al.*, 2022; Hristov, 2023a; Podlubny, 1999; Stanisławski, 2022).

In many engineering problems, there is a need to solve what's known as an 'inverse problem'. In simple terms, these issues involve identifying the input parameters of a model (e.g. boundary conditions or material parameters) based on observations (outputs) of the model. Typically, these problems are challenging because they are ill-posed (Aster *et al.*, 2013c; Kaipio and Somersalo, 2005). Examples of inverse problems in various applications are included in the articles (Montazeri *et al.*, 2022; Brociek *et al.*, 2024; Ashurov *et al.*, 2023; Wang *et al.*, 2023; Ibraheem and Hussein, 2023; Brociek *et al.*, 2023; Magacho *et al.*, 2023). For example, the article (Brociek *et al.*, 2024) focuses on an inverse problem concerning the identification of the aerothermal heating of a reusable launch vehicle based on temperature measurements taken in the thermal protection system (TPS) of this vehicle. The mathematical model of the TPS presented in the paper takes into account the dependence on temperature of the material parameters, as well as the thermal resistances occurring in the contact zones of the layers. To solve the inverse problem, the Levenberg-Marquardt method was applied.

One approach to solving inverse problems is to create a fitness function (or loss function, error function), and then optimize it to find the identified parameters. In this context, metaheuristic optimization algorithms can be very effective. In the paper (Hassan and Tallman, 2021), the authors utilize genetic algorithms, simulated annealing, and particle

swarm optimization to solve the piezoresistive inverse problem in self-sensing materials. The considered problem was ill-posed and multi-modal. The results obtained in the study indicate that the genetic algorithm proved to be the most effective. As another example, the article (Al Thobiani *et al.*, 2022) addresses an inverse problem for crack identification in two-dimensional structures. The authors utilize the eXtended Finite Element Method (XFEM) associated with the original Grey Wolf Optimization (GWO) and an improved GWO using Particle Swarm Optimization (PSO) (IGWO). The utilization of heuristic optimization algorithms for inverse problems in models with fractional derivatives can be found in papers (Brociek *et al.*, 2020; Brociek and Słota, 2015). In both of these articles, the Ant Colony Optimization (ACO) algorithm was applied. The first publication involved a comparison with an iterative method. In the second article, heat flux on the boundary was identified. Additionally, papers (Kalita *et al.*, 2023; Alyami *et al.*, 2024; Zhang and Chi, 2023) address metaheuristic optimization algorithms and their applications.

In this article, an algorithm for solving the inverse problem for the equation of anomalous diffusion is presented. This equation is a partial differential equation with fractional derivatives. The Caputo derivative was adopted as the derivative with respect to time, while the Riemann-Liouville derivative was utilized for the derivative with respect to space. In the considered inverse problem, the objective was to identify the function appearing in the fractional boundary condition as well as the orders of the derivatives. To achieve this, several metaheuristic optimization algorithms were used and compared.

2. Mathematical Model of Anomalous Diffusion with Fractional Boundary Condition

In this article, we consider a mathematical model of anomalous diffusion with fractional derivatives both in time and space. Models of this type can effectively be used to model mass and heat transport phenomena in porous media (Brociek *et al.*, 2019; Sobhani *et al.*, 2023; Kukla *et al.*, 2022). The model consists of the following fractional-order partial differential equation:

$$\frac{{}_C\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = \lambda(x,t)\frac{{}_{RL}\partial^{\beta} u(x,t)}{\partial x^{\beta}} + f(x,t), \quad x \in (0,L),\ t \in (0,T]. \tag{1}$$

The equation is supplemented with the initial condition:

$$u(x,0) = \varphi(x), \quad x \in [0,L], \tag{2}$$

and boundary conditions:

$$u(0,t) = 0, \quad t \in (0,T], \tag{3}$$

$$u(L,t) + \left(\lambda(x,t)\frac{{}_{RL}\partial^{\beta-1} u(x,t)}{\partial x^{\beta-1}}\right)\Bigg|_{x=L} = \psi(t). \tag{4}$$

It is important to note the boundary condition (4) at the right boundary of the considered domain. It takes the form of a Robin boundary condition with a fractional derivative. In the differential equation (1), two different fractional-order derivatives were assumed. The derivative of order $\alpha \in (0, 1)$ with respect to time is a Caputo-type derivative, defined by the following formula:

$$\frac{{}_C\partial^\alpha u(x,t)}{\partial t^\alpha} = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{\partial f(x,s)}{\partial s} (t-s)^{-\alpha}\, ds. \tag{5}$$

In the case of the derivative with respect to space, a fractional-order derivative of order $\beta \in (1, 2)$ Riemann-Liouville type was applied:

$$\frac{{}_{RL}\partial^\beta u(x,t)}{\partial x^\beta} = \frac{1}{\Gamma(2-\beta)} \frac{\partial^2}{\partial s^2} \int_0^x f(s,t)(t-s)^{1-\alpha}\, ds. \tag{6}$$

In the model, it is also assumed that λ is a continuous, positive function called the diffusion coefficient, and the functions f, φ, and ψ are also continuous functions. To effectively model and conduct simulations in such models, the first step is to solve equations (1)–(4). This task is known as the direct problem (or forward problem). In the next section, a numerical method to solve the considered equation is described.

3. Numerical Method of Forward Problem

This article primarily focuses on the inverse problem. The presented approach requires optimization of the fitness function. However, in the optimization process, it is necessary to repeatedly solve equations (1)–(4), that is, the so-called forward problem. To solve it, an implicit finite difference scheme is applied.

Firstly, the considered domain $\Omega = [0, T] \times [0, L]$ is discretized, resulting in a grid $S = \{(x_i, t_k)\colon i = 0, 1, \ldots, N,\ k = 0, 1, \ldots, K\}$ with steps $\Delta x = \frac{L}{N}$, $x_i = i\Delta x$, $\Delta t = \frac{T}{K}$, $t_k = k\Delta t$. Then, for all functions involved in the model, the values at the grid points S are determined. We use the following notation: $\lambda_i^k = \lambda(x_i, t_k)$, $f_i^k = f(x_i, t_k)$, $\varphi_i = \varphi(x_i)$, $\psi^k = \psi(t_k)$. Let $u_i^k = u(x_i, t_k)$ denote the values of the exact solution at the points (x_i, t_k), and U_i^k represent the corresponding values obtained from the numerical solution.

To derive the implicit finite difference scheme, we need to apply the approximation of the Riemann-Liouville derivative (6) in the form of the shifted Grünwald formula (Tian *et al.*, 2015; Tadjeran and Meerschaert, 2007):

$$\left.\frac{{}_{RL}\partial^\beta u(x,t)}{\partial x^\beta}\right|_{(x_i,t_{k+1})} \approx \frac{1}{(\Delta x)^\beta} \sum_{j=0}^{i+1} g_{\beta,j} U_{i-j+1}^{k+1}, \tag{7}$$

where $g_{\beta,j} = \frac{\Gamma(j-\beta)}{\Gamma(-\beta)\Gamma(j+1)}$. Then, we approximate the Caputo derivative (5) using the following formula (Lin and Xu, 2007):

$$\left.\frac{{}_C\partial^\alpha u(x,t)}{\partial t^\alpha}\right|_{(x_i,t_{k+1})} \approx \frac{1}{\Gamma(2-\alpha)(\Delta t)^\alpha}\sum_{j=1}^{k}\left(j^{1-\alpha}-(j-1)^{1-\alpha}\right)\left(U_i^{k-j+1}-U_i^{k-j}\right). \tag{8}$$

The derivative appearing in the boundary condition (4), after considering the zero condition at the left boundary, is approximated as follows:

$$\left.\frac{{}_{RL}\partial^{\beta-1}u(x,t)}{\partial x^{\beta-1}}\right|_{(x_N,t_{k+1})} \approx \frac{1}{(\Delta x)^{\beta-1}}\left(\sum_{j=1}^{N} g_{\beta-1,j}U_{N-j+1}^{k+1} + g_{\beta-1,0}\left(3U_N^{k+1}-3U_{N-1}^{k+1}+U_{N-2}^{k+1}\right)\right). \tag{9}$$

Combining all of the above approximations together and after appropriate transformations, we obtain the implicit difference scheme, which can be expressed in matrix form as:

$$A^1U^1 = Q^0 + F^1, \tag{10}$$

$$A^{k+1}U^{k+1} = (1-b_1)Q^k + \sum_{j=1}^{k-1}(b_j - b_{j+1})Q^{k-j} + b_kQ^0 + F^{k+1}, \tag{11}$$

where $b_j = (j+1)^{1-\alpha} - j^{1-\alpha}$. And the vectors U^k, Q^k, F^k have the following form:

$$\begin{aligned} U^k &= \left[U_1^k, U_2^k, \ldots, U_N^k\right]^T,\\ Q^k &= \left[U_1^k, U_2^k, \ldots, U_{N-1}^k, 0\right]^T,\\ F^k &= \left[(\Delta t)^\alpha\Gamma(2-\alpha)f_1^k, (\Delta t)^\alpha\Gamma(2-\alpha)f_2^k, \ldots,\right.\\ &\qquad \left.(\Delta t)^\alpha\Gamma(2-\alpha)f_{N-1}^k, (\Delta x)^{\beta-1}\psi^k\right]. \end{aligned}$$

The matrix A^k, dependent on the time step k, is of size $N \times N$. Its coefficients are determined by the following formula:

$$a_{ij}^{k}=\begin{cases}-r\lambda_i^{k+1}g_{\beta,i-j+1}, & 1\leqslant j\leqslant i-1,\ 1\leqslant i\leqslant N-1,\\ 1-r\lambda_i^{k+1}g_{\beta,1}, & 1\leqslant j=i\leqslant N-1,\\ -r\lambda_i^{k+1}g_{\beta,0}, & j=i+1,\ 1\leqslant i\leqslant N-1,\\ \lambda_N^{k+1}g_{\beta-1,N-j+1}, & 1\leqslant j\leqslant N-3,\ i=N,\\ \lambda_N^{k+1}g_{\beta-1,3}+\lambda_N^{k+1}g_{\beta-1,0}, & j=N-2,\ i=N,\\ \lambda_N^{k+1}g_{\beta-1,2}-3\lambda_N^{k+1}g_{\beta-1,0}, & j=N-1,\ i=N,\\ (\Delta x)^{\beta-1}+\lambda_N^{k+1}g_{\beta-1,1}+3\lambda_N^{k+1}g_{\beta-1,0}, & j=i=N,\\ 0, & i+2\leqslant j\leqslant N,\ 1\leqslant i\leqslant N-2.\end{cases}\tag{12}$$

In the above formula, the symbol r is defined as $r=\frac{(\Delta t)^{\alpha}\Gamma(2-\alpha)}{(\Delta x)^{\beta}}$. Equations (10)–(11) define systems of linear equations. Solving these systems will yield the approximate values of the function u at the grid points S. Article (Xie and Fang, 2020) includes theorems regarding the stability and convergence of the numerical scheme (10)–(12). In the case of the scheme under consideration, the convergence order is $O((\Delta t)^{2-\alpha}+(\Delta x)^2)$.

4. Description of Inverse Problem

In mathematical modelling, as well as in various computer simulations it is essential to use proper mathematical models. In this paper, time-space fractional diffusion equation (TSFDE) with fractional boundary contition is considered. This model was further described in Section 2 and can be used as an effective tool in modelling the heat conduction in porous materials (Brociek *et al.*, 2019). To solve model described by (1)–(4) it is necessary to know full information about the model input, such as material's parameters, geometry or model coefficients. In many engineering issues, it is impossible to have the knowledge of all model's information. It might be because of the lack of measuring equipment, or toughness in choosing the parameters. The usual problem is the choice of such entry model's parameters – input, that the model's result – output (e.g. temperature measurements in a chosen control point) takes the proper value. Problems of this sort are named inverse problems and are usually hard to solve (Özişik and Orlande, 2021; Aster *et al.*, 2013c, 2013a, 2013b). To put it simply, the problem is the identifying of the input parameters to fit to the measurement data (part of model's output) as closely as possible.

In this article, in order to solve the inverse problem, an approach is presented which involves optimizing the following fitness function:

$$\mathcal{F}(a_0,a_1,\ldots,a_{dim})=\sum_{w=1}^{W}\big(u_w(a_0,a_1,\ldots,a_{dim})-\overline{u}_w\big)^2.\tag{13}$$

Symbols $a_0,a_1,\ldots,a_{dim}$ are marked as unknown input parameters of the model-parameters, which are to be identified. Objective function $\mathcal{F}$ is dependent on these pa-

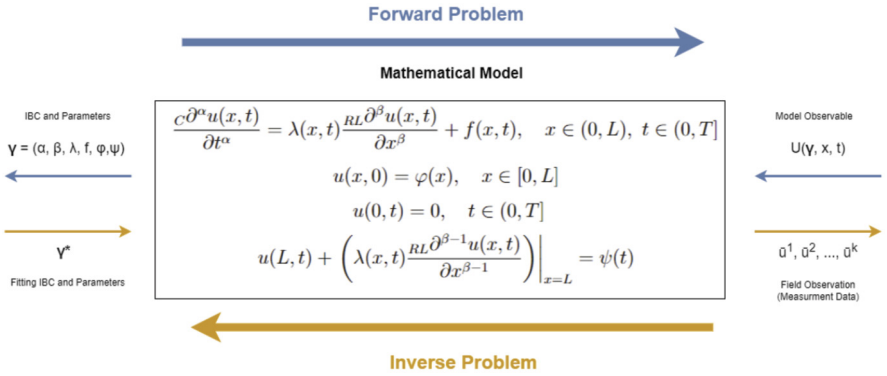

Fig. 1. Data flow diagram for forward and inverse problem.

rameters. By $dim + 1$ the size of optimization task is marked, W is a number of data in model's output (e.g. measurement data), $u_w(a_0, a_1, \ldots, a_{dim})$ $(w = 1, 2, \ldots, W)$ denotes output values calculated from the model for fixed input parameters $a_0, a_1, \ldots, a_{dim}$ and by $\overline{u}_w$ the data output (measurement data) is marked, to which model should fit itself. So, function $\mathcal{F}$ measures "how close" are the calculated values from a model (for fixed input parameters $a_0, a_1, \ldots, a_{dim}$) to output values given in a problem (e.g. measurement data). Solving given inverse problem is based on finding the minimum of fitness function relative to unknown parameters $a_0, a_1, \ldots, a_{dim}$. Hence, the use of selected metaheuristic algorithms for finding the minimum of the fitness function is justified. In Section 6, a computational example is presented and algorithms' efficiency is discussed. Figure 1 schematically presents the data flow in forward and inverse problems.

5. Metaheuristics Optimization Algorithms

Heuristic optimization algorithms for searching objective function's minimum are based on simulating group's intelligence and communication between the individuals in order to effectively search the space.

They are used for finding points in search space close to the optimal one (global minimum) in terms of fitness function. Very commonly fitness function describes a certain dependence (e.g. approximate solution error), which in an engineering problem should have the lowest possible value (problem of minimization). In contrast to classic mathematical methods, they have small requirements for objective function, which is their biggest advantage. Usually, within heuristic optimization algorithms, two phases of searching can be distinguished: exploration part – searching through possibly the vastest part of space, and exploitation part – looking for good quality solutions in a narrow part of searched space. Algorithms of this sort use different techniques, from simple local searching to advanced evolutionary processes. They use mechanisms preventing the method from getting stuck in a limited area of searched space (falling into a local minimum). These algorithms are independent of a specific problem (they do not depend on the fitness function). Algorithms

use knowledge about the problem and/or experience accumulated in a process of searching the domain (agents "communicate" with one another), with quality not decreasing as the iteration increases. This kind of algorithms falls into the category of probabilistic algorithms, meaning that their method of work include some random elements. However, a well-tuned algorithm in a vast majority of cases should be able to provide solutions, which are close to one another.

The biggest problem of heuristic optimization algorithms is tuning them properly, according to the problem to be solved. Metaheuristic optimization algorithms can be divided into four major groups:

- *Swarm Intelligence* (*SI*) *algorithms*. Inspiried by the behaviour of a swarm or a group of animals in nature. Examples: Ant Colony Optimization (ACO), Artificial Bee Colony (ACO) or Firefly Algorithm (FA).
- *Evolutionary Algorithms* (*EA*). Their description comes from natural behaviours occurring in evolutionary biology. Examples: Genetic Algorithm (GA), Differential Evolution (DE).
- *Physics-based Algorithms* (*PhA*). PhA algoritms base their descirption on physics' laws. Examples: Gravitational Search Algorithm (GSA), Electromagnetic Field Optimization (EFO).
- *Human-based algorithms*. By simulating some of natural human's behaviours, researchers proposed a few algorithms for solving optimization problems. Examples: Group Teaching Optimization Algorithm (GTOA), Collective Decision Optimization (CSO).

These algorithms gained interest because of their effectiveness in various optimization engineering problems. Examples of their usefulness include publications (Kalita *et al.*, 2023; Alyami *et al.*, 2024; Zhang and Chi, 2023; Brociek and Słota, 2015). In this paper, some of metaheuristic optimization algorithms were used and compared, from which, three best (in terms of solving inverse problem for model with fractional derivative) were described in further subsections.

5.1. *Group Teaching Optimization*

Group Teaching Optimization Algorithm (GTOA) described in this section takes inspiration from the process of group teaching. The goal of the process is to improve knowledge of the group of students. The process of teaching can be realized by different means, through learning with a teacher, exchanging knowledge between the students or self-improvement. Each student acquires knowledge with different efficiency, so it is natural to divide them into two groups: students with normal abilities and outstanding students. The teacher, in order to maximize the result, must use different methods while teaching each group. All of these mechanisms were used as an inspiration in creating Group Teaching Optimization Algorithm (Zhang and Jin, 2020).

For algorithm's presentation, the following notation is used:
dim – dimension of the task, N – number of students in population, $\mathcal{F}$ – fitness function,

$\mathbf{x}_i^t = [x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,dim}^t]$ – i-th student during iteration t before learning with a teacher,

$\mathbf{x}_{teacher}^t = [x_{teacher,1}^t, x_{teacher,2}^t, \ldots, x_{teacher,dim}^t]$ – teacher in iteration t,

$\mathbf{x}_{ALT_i}^t = [x_{ALT_i,1}^t, x_{ALT_i,2}^t, \ldots, x_{ALT_i,dim}^t]$ – i-th student after learning with teacher during iteration t,

$\mathbf{x}_{ALS_i}^t = [x_{ALS_i,1}^t, x_{ALS_i,2}^t, \ldots, x_{ALS_i,dim}^t]$ – i-th student after learning with other students during iteration t,

$\mathbf{x}_{AVG}^t = [x_{AVG,1}^t, x_{AVG,2}^t, \ldots, x_{AVG,dim}^t]$ – average students' knowledge within the population during iteration t,

$\mathbf{x}_{best}^t = [x_{best,1}^t, x_{best,2}^t, \ldots, x_{best,dim}^t]$ – the best student in iteration t,

$\mathbf{x}_{second}^t = [x_{second,1}^t, x_{second,2}^t, \ldots, x_{second,dim}^t]$ – the second best student in iteration t,

$\mathbf{x}_{third}^t = [x_{third,1}^t, x_{third,2}^t, \ldots, x_{third,dim}^t]$ – the third best student in iteration t.

To generalize, in a process of a group teaching, a few steps can be distinguished. Below, these steps are introduced along with their mathematical description, which make up a model of algorithm's operation.

- **Step 1** – *Choosing the teacher*. During this step, a so-called teacher is chosen from the whole population. The evaluation of students in a population is determined by their fitness value – the smaller the value is, the better the quality (knowledge) of the student. The process of choosing the teacher is done on the basis of the following equation:

$$\mathbf{x}_{teacher}^t = \begin{cases} \mathbf{x}_{best}^t, & \mathcal{F}(\mathbf{x}_{best}^t) < \mathcal{F}(\frac{\mathbf{x}_{best}^t + \mathbf{x}_{second}^t + \mathbf{x}_{third}^t}{3}), \\ \frac{\mathbf{x}_{best}^t + \mathbf{x}_{second}^t + \mathbf{x}_{third}^t}{3}, & \mathcal{F}(\mathbf{x}_{best}^t) \geqslant \mathcal{F}(\frac{\mathbf{x}_{best}^t + \mathbf{x}_{second}^t + \mathbf{x}_{third}^t}{3}). \end{cases} \tag{14}$$

- **Step 2** – *Division of the students*. During this step, all individuals within the population are divided into two, equally populated groups based on their knowledge (quality of their result). The results of this division are two groups, outstanding and normal groups of students.
- **Step 3** – *Learning with a teacher*. In case of learning with a teacher, the process differs for both of the groups created in the previous step. The teacher uses different methods for different students groups. Mathematically, this process can be described with following equations:
  for students in the outstanding group:

$$\mathbf{x}_{ALT_i}^t = \mathbf{x}_i^t + a\left(\mathbf{x}_{teacher}^t - fb\mathbf{x}_{AVG}^t - fc\mathbf{x}_i^t\right), \tag{15}$$

  for students in the normal group:

$$\mathbf{x}_{ALT_i}^t = \mathbf{x}_i^t + 2d\left(\mathbf{x}_{teacher}^t - \mathbf{x}_i^t\right). \tag{16}$$

  In equations (15), (16), symbols a, b, c, d were used to represent random numbers within the scope [0, 1], symbol f represents the so-called teaching factor. In this case, $f = 1$.

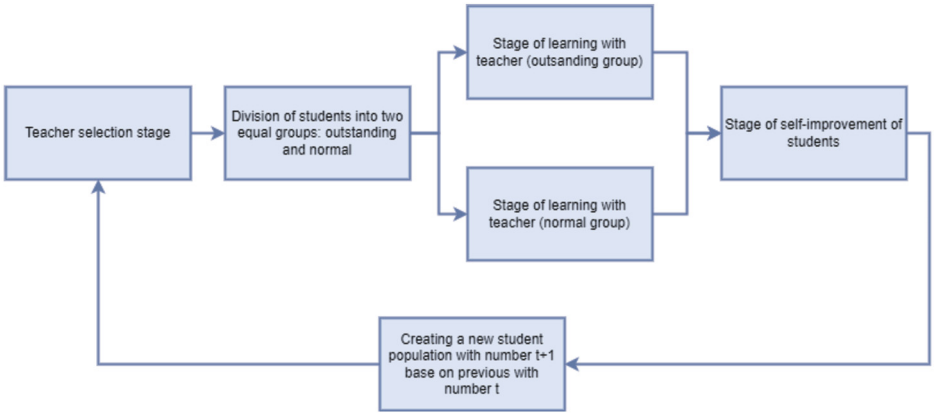

Fig. 2. Diagram of next steps of GTOA.

If student's knowledge increased, after the learning with a teacher, then the student goes to the next step. Otherwise, to the next step goes an individual from before learning with a teacher, that is:

$$\mathbf{x}_{ALT_i}^t = \begin{cases} \mathbf{x}_{ALT_i}^t, & \mathcal{F}(\mathbf{x}_{ALT_i}^t) < \mathcal{F}(\mathbf{x}_i^t), \\ \mathbf{x}_i^t, & \mathcal{F}(\mathbf{x}_{ALT_i}^t) \geqslant \mathcal{F}(\mathbf{x}_i^t). \end{cases} \tag{17}$$

- **Step 4** – *self-learning of students*. This step simulates students learning together during their free time. Students can share knowledge between one another and learn together, they can learn by themselves as well. The mathematical description of this process is as follows:

$$\mathbf{x}_{ALS,i}^t = \begin{cases} \mathbf{x}_{ALT,i}^t + e(\mathbf{x}_{ALT,i}^t - \mathbf{x}_{ALT,j}^t) + g(\mathbf{x}_{ALT,i}^t - \mathbf{x}_i^t), \\ \quad \text{for } \mathcal{F}(\mathbf{x}_{ALT,i}^t) < \mathcal{F}(\mathbf{x}_{ALT,j}^t), \\ \mathbf{x}_{ALT,i}^t - e(\mathbf{x}_{ALT,i}^t - \mathbf{x}_{ALT,j}^t) + g(\mathbf{x}_{ALT,i}^t - \mathbf{x}_i^t), \\ \quad \text{for } \mathcal{F}(\mathbf{x}_{ALT,i}^t) \geqslant \mathcal{F}(\mathbf{x}_{ALT,j}^t). \end{cases} \tag{18}$$

Symbols e, g were used to represent random number from [0, 1]. Index $j \neq i$ appearing in the equation (18) represents a random student. Hence, the interaction between students i and j. Those students, who increased their knowledge after this step, pass to the next population (denoted by $t + 1$), meaning:

$$\mathbf{x}_i^{t+1} = \begin{cases} \mathbf{x}_{ALT,i}^t, & \mathcal{F}(\mathbf{x}_{ALT,i}^t) < \mathcal{F}(\mathbf{x}_{ALS,i}^t), \\ \mathbf{x}_{ALS,i}^t, & \mathcal{F}(\mathbf{x}_{ALT,i}^t) \geqslant \mathcal{F}(\mathbf{x}_{ALS,i}^t). \end{cases} \tag{19}$$

Figure 2 schematically depicts the next steps of GTOA algoritm, while Fig. 3 presents the block diagram of the algorithm. Pseudocode 1 presents the next steps of GTOA.

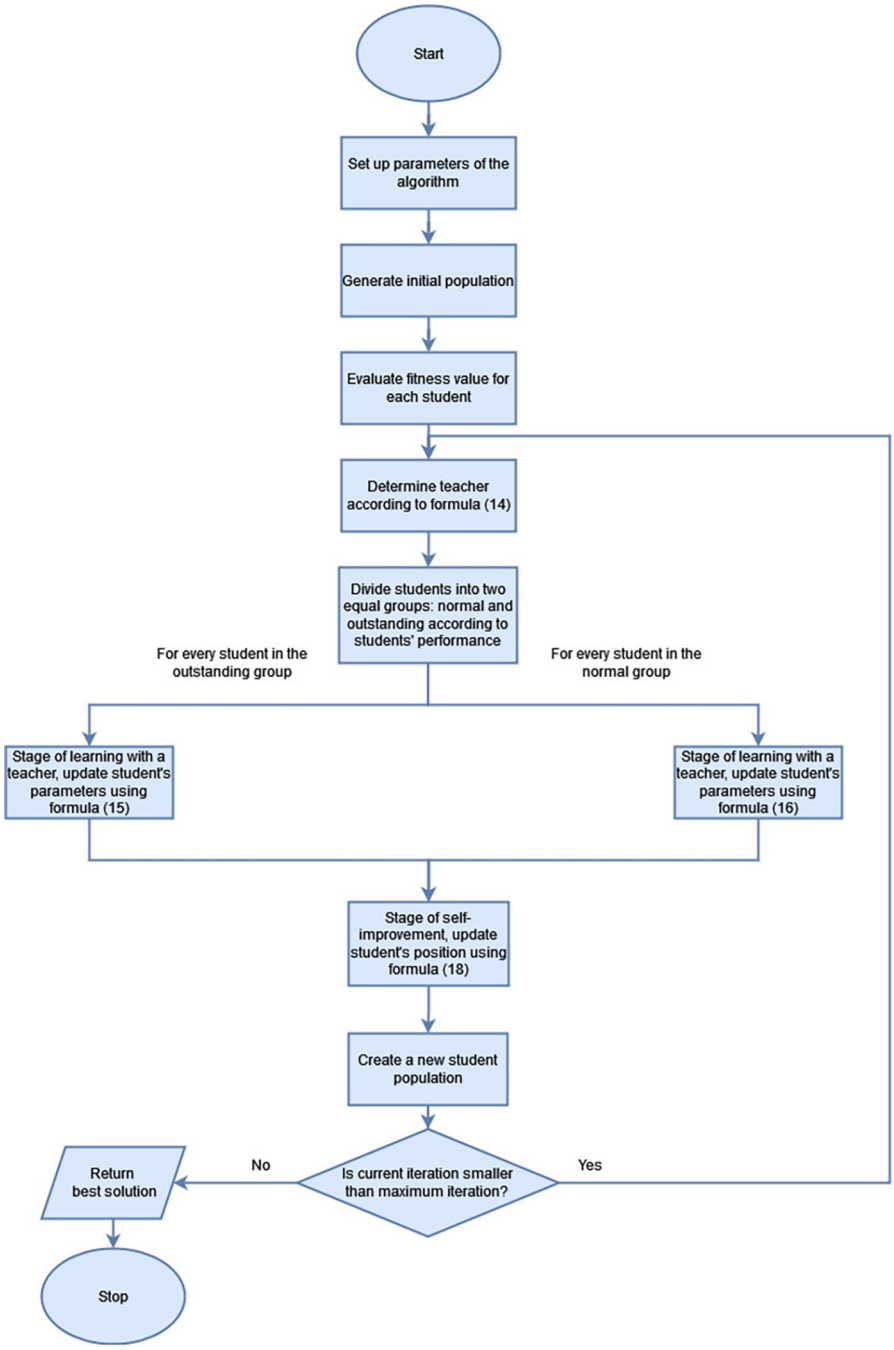

Fig. 3. Block diagram of GTOA.

Algorithm 1: Pseudocode of GTOA

Set up the parameters of the algorithm (population size, number of iterations).
Generate random distribution of agents $\mathbf{x}_i^0$ of the initial population in the search space.
Calculate the value of the fitness function for each individual in the population.
Set up counter $t = 0$.
while $t < T$ **do**
  Determine teacher $\mathbf{x}_{teacher}^t$ in t-th iteration (equation (14)).
  Divide students into two equal groups: normal and outstanding.
  for *for each student in outstanding group* **do**
    The stage of learning with the teacher. Update students' knowledge according to the equation (15). The $\mathbf{x}_{ALT_i}^t$ vectors are obtained for the outstanding group.
  end
  for *for each student in normal group* **do**
    The stage of learning with the teacher. Update students' knowledge according to the equation (16). The $\mathbf{x}_{ALT_i}^t$ vectors are obtained for the normal group.
  end
  For students in both groups, check if student's knowledge has been improves (equation (17)).
  for *for each student in population* **do**
    The stage of self improvement by students. Update students' knowledge according to the equation (18). The $\mathbf{x}_{ALS_i}^t$ vectors are obtained for each student in the population.
  end
  For students in the population, check if student's knowledge has been improved (equation (19)). A new population for the next iteration has been created.
  Increment counter $t := t + 1$.
end
Return $\mathbf{x}_{best}^T$ (the best individual in the last iteration).

5.2. *Artificial Bee Colony*

Artificial Bee Colony (ABC) is one of many metaheuristic algorithms based on animals' behaviour in their natural environment. In order to find food sources, the algorithm divides bees into two groups:

Working bees – bees that at the moment are scavenging through the already found food sources. For those bees, important factors are the distance between the source and the hive and the amount of nectar in the food source.

Unclassified bees – those bees' mission is to search for new food sources. They can be further divided into two groups: observers and scouts. Scouts look for new food sources

randomly, while observers plan their search based on the information they're provided with.

Bees exchange information by performing a special dance. Observer bees decide how to search the space based on the dance of other bees. After the collection of nectar, every bee can decide, whether they should share information about the food source they've been exploring, keep on exploring it without the information exchange with other bees or discard the food source and become an observer. In order to present the ABC algorithm, a following notation has been used:

dim – dimension of the task,

N – number of bees in a colony = number of food sources,

$\mathcal{F}$ – fitness function,

$\mathbf{x}_i^t = [x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,dim}^t]$ – i-th food source in iteration t,

$\mathbf{v}_i^t = [v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,dim}^t]$ – i-th modified food source during iteration t,

$\mathbf{p}^t = [p_1^t, p_2^t, \ldots, p_N^t]$ – probabilities of choosing food source by a bee during iteration t.

The process of an ABC algorithm can be generalized into a few steps. Below, these steps are presented together with their corresponding mathematical descriptions.

- **Step 1** – *Dance*. At this point scouting bees return to the hive and begin to share information about the food source they've been exploring. Based on the information provided by the scouts, every source is evaluated and assigned a probability according to its quality in comparison with other food sources. It is depicted by an equation below:

$$p_i = \frac{fit(x_i)}{\sum_{j=1}^{N} fit(x_j)}, \tag{20}$$

where

$$fit(x_i) = \begin{cases} \frac{1}{1+\mathcal{F}(x_i)}, & \text{if } \mathcal{F}(x_i) \geqslant 0, \\ |1 + \mathcal{F}(x_i)|, & \text{if } \mathcal{F}(x_i) < 0. \end{cases} \tag{21}$$

- **Step 2** – *Leaving the hive*. After the dance ends, every observer chooses one of the food sources and sets out to explore it (one source can be chosen multiple times), the source is being modified and if the modified source is better than the original one, it replaces the original one. The formula used to modify the food source is the following:

$$v_i^t = x_i^t + \Phi_i\left(x_i^t - x_k^t\right). \tag{22}$$

In the equation (22), Φ_i is a pseudo-random number between 0 and 1, k is a randomly selected index different than i. If the fitness value of modified food source is better than the one's before exploration, the modified one replaces the original as a food source, otherwise it is discarded.

- **Step 3** – *Abandoning old food sources*. All of the sources which haven't been chosen by any of the bees are being discarded and replaced with a new one, that is created using the following formula:

$$x_i = x_{\min} + \omega_i (x_{\max} - x_{\min}), \tag{23}$$

where ω_i is a pseudo-random number within the range [0, 1] and max, min represent the upper and lower limits of the search area, respectively.

More details about the algorithm itself and its exemplary applications are presented in Li *et al.* (2012), Karaboga and Basturk (2007), Roeva (2018).

5.3. *Jellyfish Search*

This algorithm was inspired by the behaviour of jellyfish in the ocean. It simulates factors such as following ocean currents, passive and active movements inside jellyfish swarm, time control mechanism which governs the switching between types of movement and convergence into jellyfish bloom.

Ocean Current
The direction of the ocean current is obtained by averaging all the vectors from each jellyfish in the ocean to the jellyfish that is currently in the best location:

$$\overrightarrow{trend} = X^* - e_c \frac{\sum X_i}{n_{Pop}} = X^* - e_c \mu, \tag{24}$$

where n_{Pop} is the number of jellyfish; X^* is the jellyfish currently with the best location in the swarm; e_c is the factor that governs the attraction; μ is the mean location of all jellyfish. Because we assume that there is a normal spatial distribution of jellyfish in all dimensions, the previous equation can be transformed in the following way:

$$\overrightarrow{trend} = X^* - \beta \times rand(0, 1) \times \mu, \tag{25}$$

where $\beta > 0$ is a distribution coefficient, related to the length of $\overrightarrow{trend}$. Based on the results of sensitivity analysis in numerical experiments carried out by authors of this algorithm, $\beta = 3$ is obtained.

Finally, the new location of each jellyfish is given by:

$$X_i(t+1) = X_i(t) + rand(0, 1) \times \left(X^* - \beta \times rand(0, 1) \times \mu\right). \tag{26}$$

Movement Inside Swarm
After the formation of the swarm, most jellyfish exhibit type A motion. As time goes on, more and more jellyfish begin to exhibit type B motion.

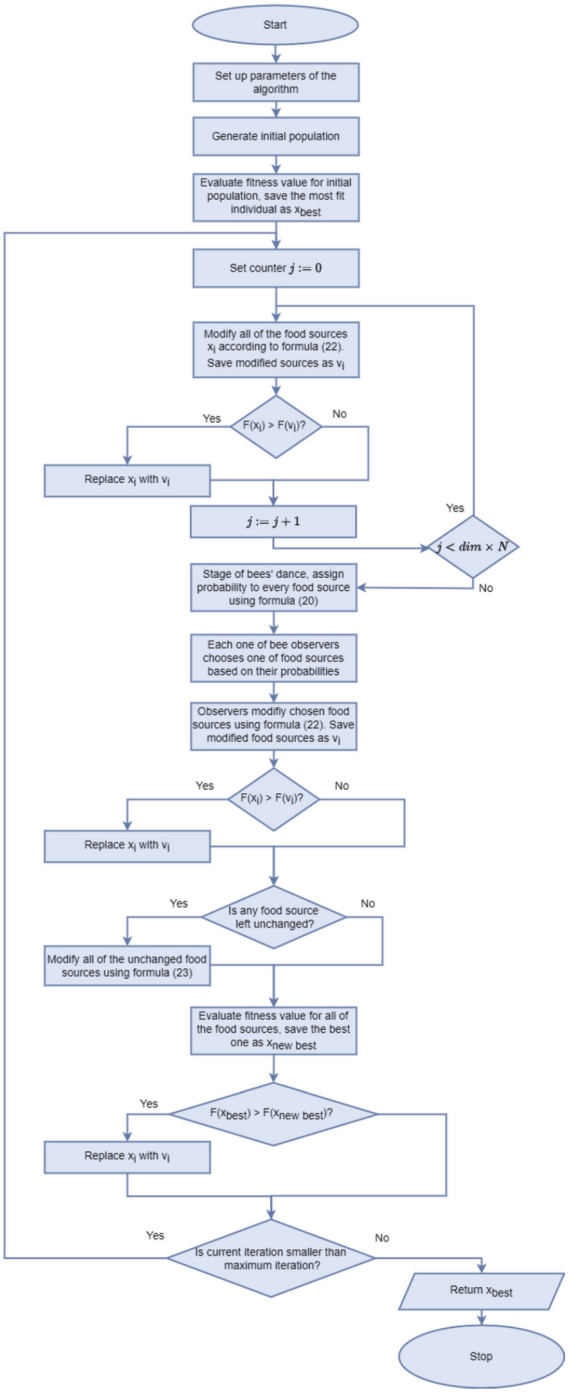

Fig. 4. Block diagram of ABC algorithm.

Algorithm 2: Pseudocode of ABC

```
Set up the parameters of the algorithm (population size, number of iterations).
Generate random distribution of agents x_i^0 of the initial population in the search space.
Calculate the value of the fitness function for each individual in the population.
Save the most fit agent as x_best.
Set up counter t = 0.
while t < T do
    Set up counter j = 0.
    while j < N × dim do
        for every food source do
            Create modified food source v_i^t using equation (22).
            if F(v_i^t) < F(x_i^t) then
                replace x_i^t with v_i^t.
            end
        end
        Increase the counter j := j + 1.
    end
    for every food source do
        Calculate probabilities p^t and assign it to all of the food sources using equation (20).
    end
    for each bee in a colony do
        Choose one food source according to the probabilities p^t, then transform the food source according to equation (22).
    end
    if any of the food sources was left unchanged then
        for every unchanged food source do
            Modify the food source using equation (23).
        end
    end
    Calculate the value of the fitness function for each individual food source, save the best fit one as x_new best
    if F(x_new best) < F(x_best) then
        Replace x_best with x_new best.
    end
    Increase the counter t := t + 1.
end
Return x_best
```

(a) Type A motions (Passive motions)

Type A motion is the motion of jellyfish around their own locations. The new location

of each jellyfish is given by

$$X_i(t+1) = X_i(t) + \gamma \times \ rand(0,1) \times (U_b - L_b), \tag{27}$$

where U_b and L_b are the upper bound and lower bound of search spaces, respectively; $\gamma = 0.1$ is a motion coefficient.

(b) Type B motions (Active motions)

To update the position of a jellyfish (i) in the type B motion, another jellyfish (j) must be selected at random. Then we compare the quantity of food at the locations of those jellyfish and create a vector from the position with less food to the position with more food. This vector is used to move jellyfish of interest (i) toward direction with more food.

$$X_i(t+1) = X_i(t) + \overrightarrow{Step}, \tag{28}$$

$$\overrightarrow{Step} = rand(0,1) \times \overrightarrow{Direction}, \tag{29}$$

$$\overrightarrow{Direction} = \begin{cases} X_j(t) - X_i(t), & \mathcal{F}(X_i) \geqslant \mathcal{F}(X_j), \\ X_i(t) - X_j(t), & \mathcal{F}(X_i) < \mathcal{F}(X_j), \end{cases} \tag{30}$$

where $\mathcal{F}$ is an fitness function of location X.

Time Control Mechanism

To regulate the movement of jellyfish between following the ocean current and moving inside the jellyfish swarm, the time control mechanism is introduced. It includes a time control function $c(t)$ which is a random value that fluctuates from 0 to 1 over time.

$$c(t) = \left| \left(1 - \frac{t}{Max_{iter}}\right) \times \big(2 \times rand(0,1) - 1\big) \right|, \tag{31}$$

where t is the time specified as the iteration number and Max_{iter} is the maximum number of iterations, which is an initialized parameter.

To decide which type of movement to use inside a swarm, the function $(1 - c(t))$ is used. When its value is less than $rand(0,1)$, the jellyfish exhibits type A motion. Otherwise, the jellyfish exhibits type B motion.

Population Initialization

In order to get initial population which is more diverse and has a lower probability of premature convergence than the one with random positions, the logistic map has been used.

$$X_{i+1} = \eta X_i (1 - X_i), \quad 0 \leqslant X_0 \leqslant 1. \tag{32}$$

X_i is the logistic chaotic value of location of the ith jellyfish; X_0 is used for generating initial population of jellyfish, $X_0 \in (0,1)$, $X_0 \notin \{0.0,\ 0.25,\ 0.75,\ 0.5,\ 1.0\}$, and parameter η is set to 4.0.

Boundary Conditions
Oceans are located around the world. The earth is approximately spherical, so when a jellyfish moves outside the bounded search area, it will return to the opposite bound.

$$X'_{i,d} = \begin{cases} (X_{i,d} - L_{b,d}) + L_b(d), & X_{i,d} > L_{b,d}, \\ (X_{i,d} - U_{b,d}) + U_b(d), & X_{i,d} < U_{b,d}. \end{cases} \tag{33}$$

$X_{i,d}$ is the location of the ith jellyfish in dth dimension; $X'_{i,d}$ is the updated location after checking boundary constraints. $U_{b,d}$ and $L_{b,d}$ are upper and lower bounds of dth dimension in search spaces, respectively.

More details about the JS and its exemplary applications are presented in the publications (Chou and Truong, 2021; Bujok, 2021; Youssef *et al.*, 2021).

Algorithm 3: Pseudocode of JS

Define fitness function $\mathcal{F}(X)$, $X = (x_1, \ldots, x_d)^T$
Set up the parameters of the algorithm (population size n_{Pop}, maximum iteration Max_{iter}).
Initialize population X_i $(i = 1, 2, \ldots, n_{Pop})$ using logistic chaotic map.
Calculate the value of the fitness function for each individual in the population.
Record the most fit agent as X^*.
Set up counter $t = 1$.
while $t \leqslant Max_{iter}$ **do**
  for $i = 1 : n_{Pop}$ **do**
    Calculate time control $c(t)$ using Eq. (31)
    if $c(t) \leqslant 0.5$ **then**
      Update position X_i using Eq. (26)
    else
      if $rand(0, 1) > (1 - c(t))$ **then**
        Update position X_i using Eq. (27)
      else
        Update position X_i using Eq. (28)
      end
    end
    Check boundary conditions using Eq. (33)
    Calculate the value of the fitness function for X_i
    if X_i *is the fittest agent* **then**
      $X^* = X_i$
    end
  end
  $t = t + 1$
end
Return $\mathbf{X}^*$

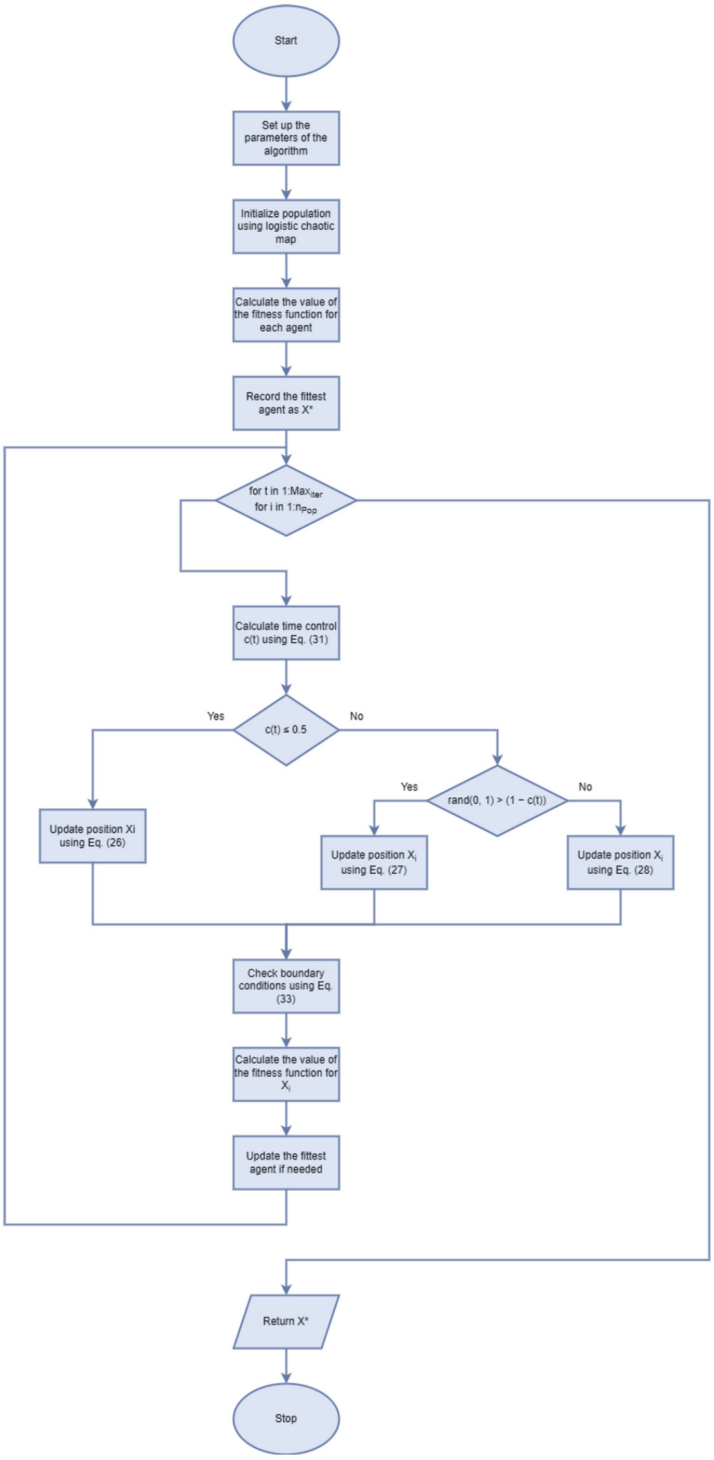

Fig. 5. Block diagram of JS.

Table 1
Reference values of sought parameters with the search domain.

Parameter	Reference value	Domain
$\alpha = a_0$	0.6	[0, 1]
$\beta = a_1$	1.5	[1, 2]
a_2	3.00901	[1, 5]
a_3	−53.1736	[−70, −20]
a_4	339.514	[250, 450]
a_5	−20	[−30, −10]
a_6	150	[50, 250]

6. Computational Example

This section is designed to present how the presented method of solving an inverse problem for the model of anomalous diffusion works. As an example, the results provided by a few chosen metaheuristic optimization algorithms were compared.

In a considered inverse problem, fractional boundary condition (4) on the right side is recreated, specifically, function ψ appearing in mentioned condition, as well as orders of fractional derivatives α, β. In the model (1)–(4), the following numeric data was assumed:

$$x \in [0, 1], \quad t \in [0, 400], \quad \lambda(x, t) = 2xt, \quad \varphi(x) = 150x,$$
$$f(x, t) = \frac{5t^{2/5}x^2}{2\Gamma\left(\frac{2}{5}\right)} - \frac{10\sqrt{\pi}x^{3/2}}{\sqrt[10]{t}\Gamma\left(\frac{9}{10}\right)} - \frac{2t\sqrt{x}(4tx - 15\pi\sqrt{tx} + 150)}{\sqrt{\pi}}.$$

The objective of this example is to test proposed algorithm (treated as a benchmark), hence the data sought – orders of derivatives α, β and function ψ are known and have the following values:

$$\alpha = 0.6, \quad \beta = 1.5, \quad \psi(t) = (\sqrt{t} - 10)^2 + \frac{2t\,(8t - 45\pi\sqrt{t} + 900)}{3\sqrt{\pi}} + 50.$$

Upon writing function ψ in a numerical form, the following was obtained:

$$3.00901t^2 - 53.1736t^{3/2} + 339.514t - 20t^{1/2} + 150.$$

So the sought function ψ and orders of derivatives α, β were identified in a following form:

$$\alpha = a_0, \quad \beta = a_1, \quad \psi(t) = a_2t^2 + a_3t^{3/2} + a_4t + a_5t^{1/2} + a_6, \tag{34}$$

where parameters $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ are unknown. In a Table 1, the domain of each parameter, as well as it's exact value are presented.

Data necessary for solving the inverse problem is value of the function u in control point $x_p = 1$ (the right boundary). This data was generated as a result of solving the

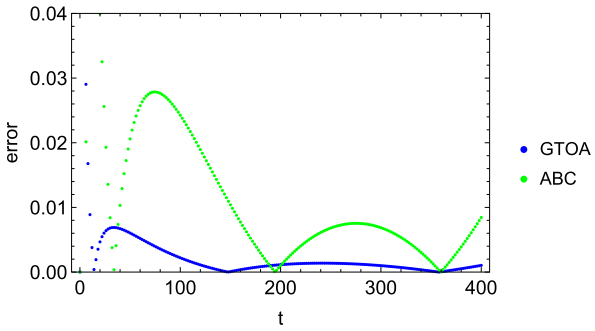

Fig. 6. Errors distribution in a control point for GTOA and ABC algorithms.

direct problem for the measurement grid 400×400. During the algorithm's work on solving the inverse problem, grid of the size 100×200 was used. Using different sizes of grid is for evading the phenomena known as inverse crime (Kaipio and Somersalo, 2005). To find the minimum of fitness function (13), which describes the error of approximated solution, selected metaheuristic optimization algorithms were used. Tested group of algorithms includes described in the Section 5 Group Teaching Optimization Algorithm (GTOA), Artificial Bee Colony (ABC) and Jellyfish Search (JS). Apart from previously mentioned, following heuristics were used: Equilibrium Optimizer (EO), Grey Wolf Optimizer (GWO), War Strategy Optimization (WSO), Tuna Swarm Optimization (TSO) and Ant Colony Optimization (ACO). In each algorithm, values of parameters such as number of iterations and size of the population were chosen in a way that ensures similar number of calls of fitness function. This is to compare the results obtained from these algorithms.

Obtained results of the search of the minimum of fitness function are presented in a Table 2. Colour blue was used to mark three algorithms (GTOA, ABC and JS), which returned a satisfying result. Colour red was used to mark the rest of the algorithms, which did not handle the task so well. Smallest value of objective function was achieved for GTOA and was approximately 0.007. Also, in this case, errors of identifying parameters $a_0, a_1, \ldots, a_6$ are the smallest. Similarly for ABC and JS, obtained errors and value of objective function are on acceptable level. It is also worth to pay attention to the results from ACO, where value of fitness function is low. However, the errors in identification of parameters are on an unacceptable level, which proves that the algorithm got stuck in a local minimum. For other algorithms,a the values of objective function are on a relatively highly level.

Another important indicator is fitness of data from control point (so called measurement data) to data obtained from model. On one hand, the measure determining this fit is the value of the objective function $\mathcal{F}$, while on the other hand, it is the errors of the reconstructed function u for the identified parameters. Figures 6 (for GTOA and ABC) and 7 (for other algorithms) present errors of distribution of the recreated function u in a control point. For the two best algorithms (GTOA and ABC), these errors are much smaller ($\approx 10^{-2}$) than for the rest of the algorithms ($\approx 10^{1}$) (see Figs. 6, 7).

Table 2
Results of parameters identification, $\overline{a}_i$ – identified value of a_i coefficient, $\Delta_{\overline{a}_i}$ – absolute error of identification, $\delta_{\overline{a}_i}$ – relative error of identification, $\mathcal{F}$ – value of fitness function.

Algorithm	$\overline{a}_i$	$\Delta_{\overline{a}_i}$	$\delta_{\overline{a}_i}[\%]$	$\mathcal{F}$
GTOA ✓✓✓	$a_0 = 0.5133$	0.0867	14.45	0.007
	$a_1 = 1.5233$	0.0233	1.55	
	$a_2 = 3.1069$	0.0978	3.25	
	$a_3 = -54.61$	1.44	2.71	
	$a_4 = 345.71$	6.19	1.82	
	$a_5 = -19.81$	0.19	0.95	
	$a_6 = 147.92$	2.08	1.38	
EO ✗	$a_0 = 0.4861$	0.1138	18.97	219.14
	$a_1 = 1.1202$	0.3797	25.31	
	$a_2 = 1.8613$	1.1476	38.14	
	$a_3 = -36.34$	16.82	31.64	
	$a_4 = 275.80$	63.70	18.76	
	$a_5 = -30.00$	10.00	50.00	
	$a_6 = 222.18$	72.18	48.12	
GWO ✗	$a_0 = 0.1282$	0.4717	78.63	217.17
	$a_1 = 1.1339$	0.3660	24.41	
	$a_2 = 1.9636$	1.0453	34.74	
	$a_3 = -39.20$	13.97	26.27	
	$a_4 = 292.47$	47.04	13.85	
	$a_5 = -16.64$	3.35	16.79	
	$a_6 = 75.69$	74.30	49.53	
WSO ✗	$a_0 = 0.7993$	0.1993	33.22	42.16
	$a_1 = 1.1456$	0.3543	23.62	
	$a_2 = 2.0023$	1.0066	33.45	
	$a_3 = -39.55$	13.62	25.61	
	$a_4 = 290.64$	48.86	14.39	
	$a_5 = -13.73$	6.26	31.34	
	$a_6 = 121.65$	28.34	18.89	
TSO ✗	$a_0 = 0.4344$	0.1655	27.59	122.48
	$a_1 = 1.1342$	0.3657	24.38	
	$a_2 = 1.9612$	1.0477	34.82	
	$a_3 = -39.04$	14.12	26.56	
	$a_4 = 291.17$	48.34	14.23	
	$a_5 = -15.54$	4.46	22.27	
	$a_6 = 95.21$	54.79	36.53	
ACO ✗	$a_0 = 0.1126$	0.4873	81.22	0.2224 (local minimum)
	$a_1 = 1.1014$	0.3985	26.57	
	$a_2 = 1.9438$	1.0652	35.41	
	$a_3 = -38.57$	14.60	27.46	
	$a_4 = 287.60$	51.91	15.29	
	$a_5 = -15.61$	4.39	21.96	
	$a_6 = 146.68$	3.31	2.21	

(*continued on next page*)

Table 2
(*continued*)

Algorithm	$\overline{a}_i$	$\Delta_{\overline{a}_i}$	$\delta_{\overline{a}_i}[\%]$	$\mathcal{F}$
	$a_0 = 0.7239$	0.1239	20.66	
	$a_1 = 1.5089$	0.0089	0.59	
	$a_2 = 3.2198$	0.2108	7.01	
ABC ✓✓	$a_3 = -56.31$	3.13	5.91	2.01
	$a_4 = 353.91$	14.39	4.24	
	$a_5 = -30$	10	50	
	$a_6 = 165.25$	15.25	10.16	
	$a_0 = 0.4684$	0.1315	21.92	
	$a_1 = 1.4316$	0.0683	4.55	
	$a_2 = 2.5251$	0.4838	16.08	
JS ✓	$a_3 = -46.30$	6.87	12.92	20.85
	$a_4 = 311.36$	28.15	8.29	
	$a_5 = -17.50$	2.49	12.46	
	$a_6 = 136.24$	13.75	9.17	

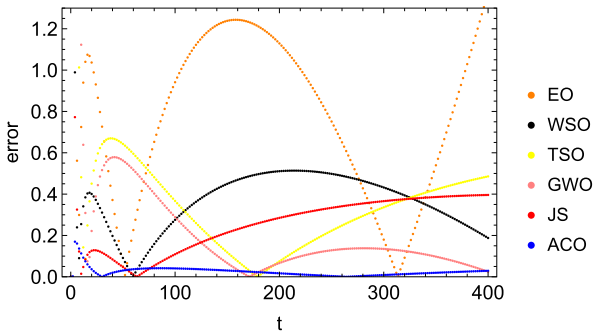

Fig. 7. Error distribution in a control point for EO, WSO, TSO, GWO, JS, ACO algorithms.

Because the exact solution is known, we decided to calculate the errors of reconstruction function u in a full domain (not only in control point). In Fig. 8 errors of reconstruction function u for identified model parameters are presented. For three best algorithms, these errors are on an acceptable level and they share similar characteristics. It is the case of GTOA (maximum error $\approx$ 1.5), ABC (maximum error $\approx$ 10) and JS (maximum error $\approx$ 20). In the rest of algorithms, maximum errors are high and unacceptable.

As part of the conducted computations, a comparison of the error of identification of the function ψ occurring in the boundary condition was also performed. These errors were calculated using the formula:

$$E_{abs} = \frac{1}{t^*} \int_0^{t^*} \left| \psi(t) - \psi_{approx}(t) \right| dt, \tag{35}$$

$$E_{rel} = E_{abs} \left(\frac{1}{t^*} \int_0^{t^*} \left| \psi(t) \right| dt \right)^{-1} 100\%, \tag{36}$$

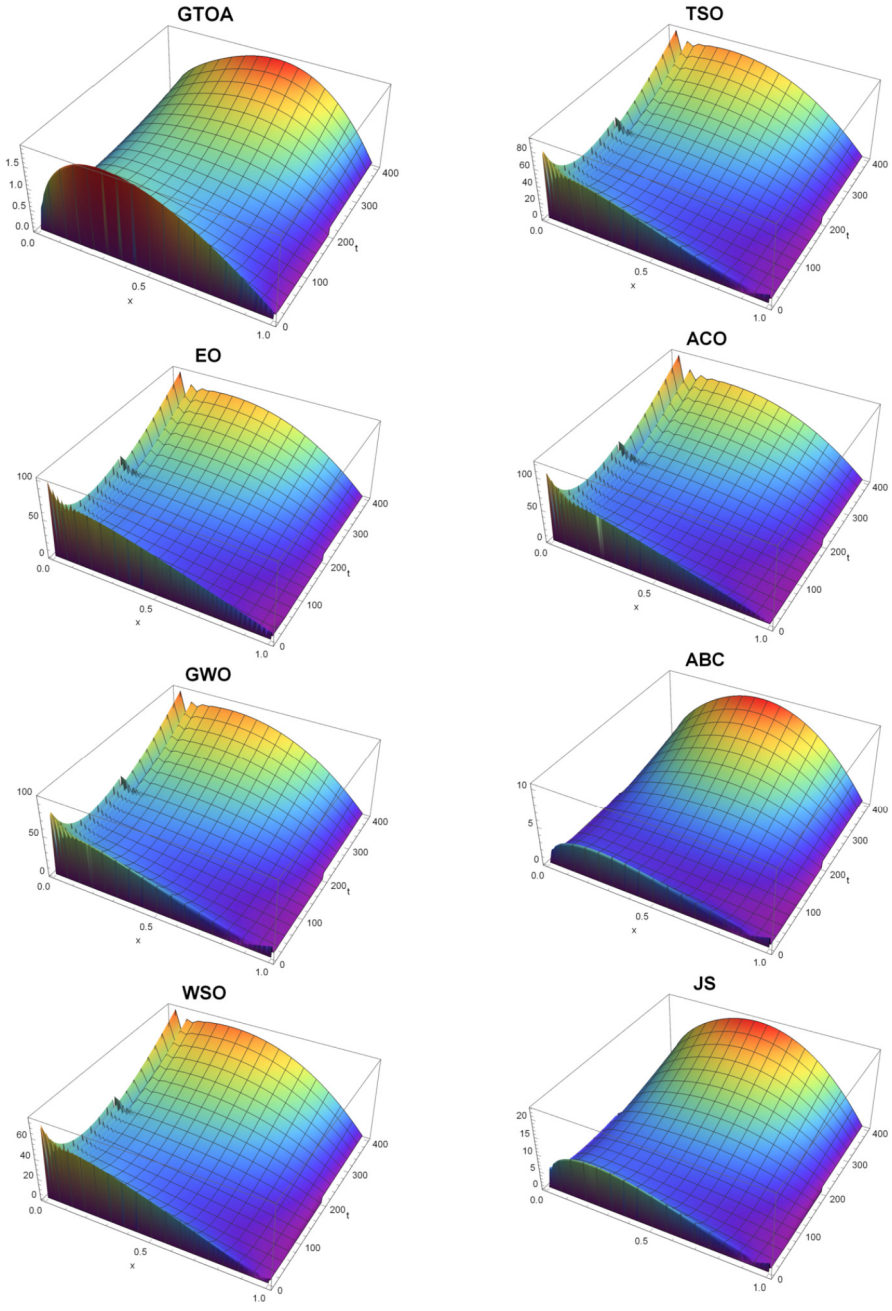

Fig. 8. Function u's reconstruction's error in a full domain.

where ψ is the exact solution, and ψ_{approx} is the approximate solution. The corresponding results are presented in Table 3. For the two best algorithms, the percentage errors are

Table 3
Absolute and relative errors of function reconstruction ψ.

Algorithm	Absolute error	Relative error [%]
GTOA	1864.98	3.21
ABC	3966.77	6.82
EO	20147.81	34.67
WSO	19815.22	34.09
TSO	20319.82	34.96
JS	9424.24	16.21
GWO	20478.45	35.23
ACO	20407.33	35.11

3.21% for GTOA and 6.82% for ABC. The JS algorithm ranked third with an error of 16.21%, while for the remaining algorithms, the errors were large $\approx 35\%$.

7. Conclusion

This article focuses on the method of solving the inverse problem for diffusion equation with fractional derivatives. In the considered task, the orders of derivatives and the function occurring in the boundary condition were identified. In the presented approach, the forward problem was solved using an implicit finite difference scheme, while the inverse problem was solved using heuristic optimization algorithms. The inverse problem turned out to be difficult to solve and required identification of seven parameters.

To solve the inverse problem, the following metaheuristic algorithms were compared: GTOA, ABC, ACO, EO, GWO, JS, TSO, WSO. Satisfactory results were obtained for the *Group Teaching Optimization Algorithm* (GTOA) and the *Artificial Bee Colony* (ABC) method. Additionally, the *Jellyfish Search* (JS) algorithm yielded an acceptable result. The remaining algorithms proved unsuitable for this type of problem.

One of the conclusions from the conducted research and next step in the research is the possibility to build a hybrid algorithm. Firstly, a heuristic algorithm could be used for initial solution localization (*exploration part*), while deterministic methods such as Nelder-Mead or Hooke-Jeeves could be employed for more focused searches (*eploitation part*). This is the next step and a plan for future research in the work carried out by the authors of this paper.

References

Al Thobiani, F., Khatir, S., Benaissa, B., Ghandourah, E., Mirjalili, S., Abdel Wahab, M. (2022). A hybrid PSO and Grey Wolf Optimization algorithm for static and dynamic crack identification. *Theoretical and Applied Fracture Mechanics*, 118, 103213. https://doi.org/10.1016/j.tafmec.2021.103213.

Alyami, M., Khan, M., Javed, M.F., Ali, M., Alabduljabbar, H., Najeh, T., Gamil, Y. (2024). Application of metaheuristic optimization algorithms in predicting the compressive strength of 3D-printed fiber-reinforced concrete. *Developments in the Built Environment*, 17, 100307. https://doi.org/10.1016/j.dibe.2023.100307.

Arul, R., Karthikeyan, P., Karthikeyan, K., Geetha, P., Alruwaily, Y., Almaghamsi, L., El-hady, E.-S. (2022). On Ψ-Hilfer fractional integro-differential equations with non-instantaneous impulsive conditions. *Fractal and Fractional*, 6(12). https://doi.org/10.3390/fractalfract6120732.

Ashurov, R., Kadirkulov, B., Ergashev, O. (2023). Inverse problem of Bitsadze–Samarskii type for a two-dimensional parabolic equation of fractional order. *Journal of Mathematical Sciences (United States)*, 274(2), 172–185. https://doi.org/10.1007/s10958-023-06587-8.

Aster, R.C., Borchers, B., Thurber, C.H. (2013a). Chapter four – Tikhonov regularization. In: Aster, R.C., Borchers, B., Thurber, C.H. (Eds.), *Parameter Estimation and Inverse Problems*, second ed. Academic Press, Boston, pp. 93–127. 978-0-12-385048-5. https://doi.org/10.1016/B978-0-12-385048-5.00004-5.

Aster, R.C., Borchers, B., Thurber, C.H. (2013b). Chapter ten – Nonlinear inverse problems. In: Aster, R.C., Borchers, B., Thurber, C.H. (Eds.), *Parameter Estimation and Inverse Problems*, second ed. Academic Press, Boston, pp. 239–252. 978-0-12-385048-5. https://doi.org/10.1016/B978-0-12-385048-5.00010-0.

Aster, R.C., Borchers, B., Thurber, C.H. (2013c). Chapter three – Rank deficiency and Ill-conditioning. In: Aster, R.C., Borchers, B., Thurber, C.H. (Eds.), *Parameter Estimation and Inverse Problems*, second ed. Academic Press, Boston, pp. 55–91. 978-0-12-385048-5. https://doi.org/10.1016/B978-0-12-385048-5.00003-3.

Bhangale, N., Kachhia, K.B., Gómez-Aguilar, J.F. (2023). Fractional viscoelastic models with Caputo generalized fractional derivative. *Mathematical Methods in the Applied Sciences*, 46(7), 7835–7846. https://doi.org/10.1002/mma.7229.

Bohaienko, V., Gladky, A. (2023). Modelling fractional-order moisture transport in irrigation using artificial neural networks. *SeMA Journal*, 81, 219–233. https://doi.org/10.1007/s40324-023-00322-8.

Brociek, R., Słota, D. (2015). Application of intelligent algorithm to solve the fractional heat conduction inverse problem. *Communications in Computer and Information Science*, 538, 356–365. https://doi.org/10.1007/978-3-319-24770-0_31.

Brociek, R., Słota, D., Król, M., Matula, G., Kwaśny, W. (2017). Modeling of heat distribution in porous aluminum using fractional differential equation. *Fractal and Fractional*, 1(1), 17.

Brociek, R., Słota, D., Król, M., Matula, G., Kwaśny, W. (2019). Comparison of mathematical models with fractional derivative for the heat conduction inverse problem based on the measurements of temperature in porous aluminum. *International Journal of Heat and Mass Transfer*, 143, 118440.

Brociek, R., Chmielowska, A., Słota, D. (2020). Comparison of the probabilistic ant colony optimization algorithm and some iteration method in application for solving the inverse problem on model with the caputo type fractional derivative. *Entropy*, 22(5), 555. https://doi.org/10.3390/E22050555.

Brociek, R., Wajda, A., Capizzi, G., Słota, D. (2023). Parameter estimation in the mathematical model of bacterial colony patterns in symmetry domain. *Symmetry*, 15(4), 782. https://doi.org/10.3390/sym15040782.

Brociek, R., Hetmaniok, E., Napoli, C., Capizzi, G., Słota, D. (2024). Identification of aerothermal heating for thermal protection systems taking into account the thermal resistance between layers. *International Journal of Heat and Mass Transfer*, 218, 124772. https://doi.org/10.1016/j.ijheatmasstransfer.2023.124772.

Bujok, P. (2021). Three steps to improve jellyfish search optimiser. *MENDEL*, 27(1), 29–40. https://doi.org/10.13164/mendel.2021.1.029.

Chou, J.-S., Truong, D.-N. (2021). A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation*, 389, 125535. https://doi.org/10.1016/j.amc.2020.125535.

Ciesielski, M., Grodzki, G. (2023). Numerical Approximations of the Riemann–Liouville and Riesz fractional integrals. *Informatica*, 389(1), 21–46 . https://doi.org/10.15388/23-INFOR540.

Hassan, H., Tallman, T.N. (2021). A comparison of metaheuristic algorithms for solving the piezoresistive inverse Problem in self-sensing materials. *IEEE Sensors Journal*, 21(1), 659–666. https://doi.org/10.1109/JSEN.2020.3014554.

Hou, J., Meng, X., Wang, J., Han, Y., Yu, Y. (2023). Local error estimate of an L1-finite difference scheme for the multiterm two-dimensional time-fractional reaction-diffusion equation with robin boundary conditions. *Fractal and Fractional*, 7(6), 453. https://doi.org/10.3390/fractalfract7060453.

Hristov, J. (2023a). Chapter 11 – Fourth-order fractional diffusion equations: constructs and memory kernel effects. In: Baleanu, D., Balas, V.E., Agarwal, P. (Eds.), *Fractional Order Systems and Applications in Engineering*, Advanced Studies in Complex Systems. Academic Press, pp. 199–214. 978-0-323-90953-2. https://doi.org/10.1016/B978-0-32-390953-2.00019-0.

Hristov, J. (2023b). Special issue "Trends in fractional modelling in science and innovative technologies". *Symmetry*, 15(4), 884. https://doi.org/10.3390/sym15040884.

Ibraheem, Q.W., Hussein, M.S. (2023). Determination of time-dependent coefficient in time fractional heat equation. *Partial Differential Equations in Applied Mathematics*, 7(12), 100492. https://doi.org/10.1016/j.padiff.2023.100492.

Ionescu, C., Lopes, A., Copot, D., Machado, J.A.T., Bates, J.H.T. (2017). The role of fractional calculus in modeling biological phenomena: a review. *Communications in Nonlinear Science and Numerical Simulation*, 51, 141–159. https://doi.org/10.1016/j.cnsns.2017.04.001.

Kaipio, J., Somersalo, E. (2005). *Statistical and Computational Inverse Problems*. Springer, New York.

Kalita, K., Ganesh, N., Shankar, R., Chakraborty, S. (2023). A fuzzy MARCOS-based analysis of dragonfly algorithm variants in industrial optimization problems. *Informatica*, 35(1), 155–178. https://doi.org/10.15388/23-INFOR538.

Karaboga, D., Basturk, B. (2007). Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (Eds.), *Foundations of Fuzzy Logic and Soft Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 789–798. 978-3-540-72950-1.

Koleva, M.N., Poveschenko, Y., Vulkov, L.G. (2021). Numerical simulation of thermoelastic nonlinear waves in fluid saturated porous media with non-local Darcy law. In: Dimov, I., Fidanova, S. (Eds.), *Advances in High Performance Computing*. Springer International Publishing, Cham, pp. 279–289. 978-3-030-55347-0.

Kukla, S., Siedlecka, U., Ciesielski, M. (2022). Fractional order dual-phase-lag model of heat conduction in a composite spherical Medium. *Materials*, 15(20), 7251.

Li, G., Niu, P., Xiao, X. (2012). Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. *Applied Soft Computing*, 12(1), 320–332. https://doi.org/10.1016/j.asoc.2011.08.040.

Lin, Y., Xu, C. (2007). Finite difference/spectral approximations for the time-fractional diffusion equation. *Journal of Computational Physics*, 225(2), 1533–1552. https://doi.org/10.1016/j.jcp.2007.02.001.

Magacho, E.G., Jorge, A.B., Gomes, G.F. (2023). Inverse problem based multiobjective sunflower optimization for structural health monitoring of three-dimensional trusses. *Evolutionary Intelligence*, 16(1), 247–267. https://doi.org/10.1007/s12065-021-00652-4.

Montazeri, M., Mohammadiun, H., Mohammadiun, M., Bonab, D., Hossein, M., Vahedi, M. (2022). Inverse estimation of time-dependent heat flux in stagnation region of annular jet on a cylinder using Levenberg–Marquardt method. *Iranian Journal of Chemistry and Chemical Engineering*, 41(3), 971–988. https://doi.org/10.30492/ijcce.2021.131987.4263.

Obembe, A.D., Al-Yousef, H.Y., Hossain, M.E., Abu-Khamsin, S.A. (2017). Fractional derivatives and their applications in reservoir engineering problems: a review. *Journal of Petroleum Science and Engineering*, 157, 312–327. https://doi.org/10.1016/j.petrol.2017.07.035.

Özişik, M.N., Orlande, H.R.B. (2021). *Inverse Heat Transfer Fundamentals and Applications*. Taylor and Francis Group.

Podlubny, I. (1999). *Fractional Differential Equations*. Academic Press, San Diego.

Roeva, O. (2018). Application of artificial Bee Colony Algorithm for model parameter identification. In: Zelinka, I., Vasant, P., Duy, V.H., Dao, T.T. (Eds.), *Innovative Computing, Optimization and Its Applications: Modelling and Simulations*. Springer International Publishing, Cham, pp. 285–303. 978-3-319-66984-7. https://doi.org/10.1007/978-3-319-66984-7_17.

Sobhani, H., Azimi, A., Noghrehabadi, A., Mozafarifard, M. (2023). Numerical study and parameters estimation of anomalous diffusion process in porous media based on variable-order time fractional dual-phase-lag model. *Numerical Heat Transfer, Part A: Applications*, 83(7), 679–710. https://doi.org/10.1080/10407782.2022.2157915.

Sowa, M., Łukasz Majka, Wajda, K. (2023). Excitation system voltage regulator modeling with the use of fractional calculus. *AEU – International Journal of Electronics and Communications*, 159, 154471. https://doi.org/10.1016/j.aeue.2022.154471.

Stanisławski, R. (2022). Fractional systems: state-of-the-art. *Studies in Systems, Decision and Control*, 402, 3–25. https://doi.org/10.1007/978-3-030-89972-1_1.

Tadjeran, C., Meerschaert, M.M. (2007). A second-order accurate numerical method for the two-dimensional fractional diffusion equation. *Journal of Computational Physics*, 220(2), 813–823. https://doi.org/10.1016/j.jcp.2006.05.030.

Tian, W., Zhou, H., Deng, W. (2015). A class of second order difference approximations for solving space fractional diffusion equations. *Mathematics of Computation*, 84(294), 1703–1727.

Wang, S., Li, Y., Zhou, Y., Peng, G., Xu, W. (2023). Identification of thermal conductivity of transient heat transfer systems based on an improved artificial fish swarm algorithm. *Journal of Thermal Analysis and Calorimetry*, 148(14), 6969–6987. https://doi.org/10.1007/s10973-023-12182-5.

Xie, C., Fang, S. (2020). Finite difference scheme for time-space fractional diffusion equation with fractional boundary conditions. *Mathematical Methods in the Applied Sciences*, 43(6), 3473–3487. https://doi.org/10.1002/mma.6132.

Youssef, H., Hassan, M.H., Kamel, S., Elsayed, S.K. (2021). Parameter estimation of single phase transformer using Jellyfish search optimizer algorithm. In: *2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, pp. 1–4. https://doi.org/10.1109/ICAACCA51523.2021.9465279.

Zhang, Y., Jin, Z. (2020). Group teaching optimization algorithm: a novel metaheuristic method for solving global optimization problems. *Expert Systems with Applications*, 148, 113246. https://doi.org/10.1016/j.eswa.2020.113246.

Zhang, Y., Chi, A. (2023). Group teaching optimization algorithm with information sharing for numerical optimization and engineering optimization. *Journal of Intelligent Manufacturing*, 1547–1571. https://doi.org/10.1007/s10845-021-01872-2.

R. Brociek obtained the MSc degree in mathematics from the Silesian University of Technology (in 2013) and the PhD in technical sciences from Czestochowa University (in 2019). He is an adjunct professor at the Department of Mathematics Applications and Methods for Artificial Intelligence, Silesian University of Technology, Gliwice, Poland. His research interests include artificial intelligence, application of computational methods to various problems in engineering and mathematical simulation. He has experience in mathematical modelling, applying of fractional calculus in engineering, as well as the application of artificial intelligence methods in optimization problems.

M. Goik is a student at the Faculty of Applied Mathematics at the Silesian University of Technology. He is currently employed as a software developer at a company that specializes in industrial automation systems. His interests include algorithms, artificial intelligence, and embedded systems. During his free time, he enjoys participating in coding competitions and staying up-to-date with the latest advancements in technology.

J. Miarka is a sophomore majoring in computer science at the Faculty of Applied Mathematics at the Silesian University of Technology. He is a participant of the Silesian University of Technology's mentoring program. His research interests include: practical application of mathematics, automation and optimization problems and machine learning.

M. Pleszczyński received the MSc degree in mathematics and the PhD degree in applied sciences, in the area of computer science from the Czestochowa University of Technology, Czestochowa, Poland, in 2001 and 2009, respectively. He is an adjunct professor with the Faculty of Applied Mathematics, Silesian University of Technology. He has authored/coauthored more than 30 research papers in international conferences and journals in the area of applied computing. He is currently working on numerical methods, particularly, by applying mathematics, computer tomography.

C. Napoli is an associate professor with the Department of Computer, Control, and Management Engineering "Antonio Ruberti", Sapienza University of Rome, since 2019, where he also collaborates with the department of Physics and the Faculty of Medicine and Psychology, as well as holding the office of scientific director of the International School of Advanced and Applied Computing (ISAAC). He received the BSc degree in physics from the Department of Physics and Astronomy, University of Catania, in 2010, where he also got the MSc degree in astrophysics in 2012 and the PhD in computer science in 2016 from the Department of Mathematics and Computer Science. Christian Napoli has been a research associate with the Department of Mathematics and Computer Science, University of Catania, from 2018 to 2019, and, previously, a research fellow and an adjunct professor with the same department from 2015 to 2018. He has been a student research fellow with the Department of Electrical, Electronics, and Informatics Engineering, University of Catania, from 2009 to 2016, a collaborator of the Astrophysical Observatory of Catania and the National Institute for Nuclear Physics, since 2010. He has been invited as a professor to the Silesian University of Technology several times, a visiting academic at the New York University, and responsible of many different institutional topics from 2011 until now for undegraduate, graduate and PhD students in computer science, computer engineering and electronics engineering. His teaching activity focused on artificial intelligence, neural networks, machine learning, computing systems, computer architectures, distributed systems, and high performance computing. He is involved in several international research projects, serves as a reviewer and member of the board program committee for major international journals and international conferences. His current research interests include neural networks, artificial intelligence, human-computer interaction and computational neuropsychology.