Web service for solving optimisation problems using swarm intelligence algorithms

Yuriy Tryus[1], *Andrii* Geiko[1], and *Grygoriy* Zaspa[1,*]

[1]Cherkasy State Technological University, Computer Science and Information Technology Department, 460 Shevchenko Blvd, 18006, Cherkasy, Ukraine

Abstract. In this research the web service which provides a user a possibility to use swarm optimisation algorithms for solving continuous and discrete extreme problems online was created. The web service includes facilities for data input and making numerical experiments related to optimisation problems research with swarm intelligence methods. For swarm intelligence algorithms numerical experiments, the user can tune the following parameters: problem dimension; swarm parts number; inertial, cognitive, and social coefficients. The user can also select algorithm stop criterion and set the number of algorithm iterations. During the optimisation problem solving, the iteration process graphical visualisation is done. Also, the protocol is being formed and it can be exported to .xls file. Checking the algorithms in the web service was made on well-known test functions and conditional and non-conditional optimisation problems; some of them are built into the service.

1 Introduction

Solving real optimisation problems existing in different areas of science, technology, or economics is complicated with existence the following properties of their mathematical models: search surface complex topology, multi-extrema, non-smoothness, multiple dimensions. In this case, using natural processes imitation based methods, which are borrowed from nature, and realise adaptive randomness of a search is expedient. The methods based on self-organised population systems collective behaviour modelling are among this kind of methods. For instance, an are ant colony optimisation (A. Colorni, M. Dorigo, V. Maniezzo (1991)), particle swarm optimisation (PSO) (J. Kennedy, R. Eberhart (1995)), grey wolf algorithms (Mirjalili S. (1995)), bee colony optimisation (D. Karaboga (2005)), glow-worm swarm optimisation (K. N. Krishnanand, D. Ghose (2005), monkey search algorithm (A. Mucherino, O. Seref (2007)), school of fish algorithm (Bastos-Filho C.J.A., de Lima Neto F.B., Lins A.J.C.C., Nascimento A.I.S., Lima M.P. (2008)), cuckoo search algorithm (X.-S. Yang, S. Deb (2009)), bats gang algorithm (Yang X. S. (2010)), bacterial optimisation (Niu Ben, Wang Hong (2012)) etc. The primary algorithm between them is particle swarm optimisation (PSO) offered by Kennedy J. i Eberhart R. in [1] and developed by many other researchers (see, for instance, [2]-[6]). It is popular, primarily because it can be used for effective solving the wide range of optimisation problems including continuous, discrete, binary, and multi-criteria optimisation.

2 Problem topicality

The swarm intelligence methods are diverse and can be used for solving problems in different domains: telecommunication, aviation, space research, robotics, medical care, biology etc. That is why swarm intelligence and its researching methods are a topical scientific direction. In many universities over the world, students of mathematical and computer faculties study PSO algorithms in courses related to the modern optimisation and operation research methods. Using these methods for solving real optimisation problems, computer mathematical software such as MatlabR2017a can be used. It has *particleswarm* function in Global Optimisation Toolbox [7]. It realises PSO method. But Matlab is a proprietary desktop application and is not available for everybody. So, creating a system which would provide the users the possibility to use PSO algorithms for solving training and practical optimisation problems online and include the theoretical information and facilities for making experiments for researching and developing PSO algorithms is a very topical problem.

The research objectives are a) reasoning of necessity of web service for solving optimisation problems of different type online with swarm intelligence methods development; b) requirements analysis to the main information technologies being used on the way of the web service creation; c) the web resource design and creation; d) making numeric experiment for testing the methods realised using test functions.

* Corresponding author: g.zaspa@chdtu.edu.ua

2.1 Problem definition

A web resource being called PSO Service should implement the main algorithms of swarm intelligence specifically canonical PSO algorithm, its adaptive and hybrid variants for solving continuous and discrete optimisation problems, one- and multi-criteria extremum problems. The web service user should have the possibility to a) enter objective function, constrain functions, direct constraint of variable values; b) set parameters for problems solving specifically: swarm dimension, search margins, calculation precision, relations type between swarm particles ("star", "ring", "random"); c) get problem solution in view understandable for the user. Also, the user should have the possibility to see the function value for every iteration. The service should visualise the PSO algorithms iterations for two-dimensional optimisation problems. For solving conditional optimisation problems, specifically mathematical programming the penalty function method with possible choosing among different penalties and setting their options will be used. For making numerical experiments on PSO algorithms ability to work and effectiveness the service should include a set of known test functions. Besides, the service should be multilingual, include general information on PSO algorithms, and their use for solving different kinds of optimisation problems.

3 PSO algorithms realised in PSO Service

Here, the non-conditional global optimisation problem should be solved, formulated as the problem objective function *f(X)* in *D* search region minimisation:

$$f(x) \to \min,\ x \in D \subset R^d, \qquad (1)$$

where D is hypercube of d dimension, $x - f$ function vector argument; x^* – objective function $f(x)$ global minimum point.

Brief consideration of the PSO algorithms of solving problem of kind (1) which are realised in the PSO Service web service is presented.

3.1 Canonical PSO algorithm for continuous optimisation problems

In canonical PSO particles swarm is a set of decision points which transfer in space searching for global optimum. While moving the particles try to improve the value found before and exchange the information with their neighbours.

The set of swarm particle positions is defined as:

$$X = \{x_1, x_2, ..., x_s\}, \qquad (2)$$

where s is particles number in a swarm.

The position of particle i is the set of its coordinates ($x_{i1}, x_{i2}, \ldots, x_{id}$) in search region D of dimension d, $i = \overline{1,s},$. For optimisation 20-50 particles are usually enough [4]. The swarm allows to find global optimum even it has particles number less than search region dimension d.

In the beginning of PSO algorithm work the particle swarm is initialised the random way. If there is no any prior information about the function being optimised the simplest way to generate the initial particles coordinates is to use the following formula:

$$x_{ij} = rand(x_{j\min}, x_{j\max}), \qquad (3)$$

where x_{ij} is coordinate number j of the particle number i, $i = \overline{1,s},\ j = \overline{1,d},$;

$rand(x_{jmin}, x_{jmax})$ is a random number with uniform distribution on interval defined with search region borders for coordinate j.

The set of particles speed vectors is also associated with the swarm:

$$V = \{v_1, v_2, ..., v_s\}, \qquad (4)$$

All speeds can be considered as equal to 0 on the initial stage. But the practice shows that the following formula gives the best results [3]:

$$v_{ij} = [rand(x_{j\min}, x_{j\max}) - x_{ij}]/2, \qquad (5)$$

where v_{ij} – speed component number j of particle i.

This way of defining the initial speeds guarantees that no particle gets out of search region in the next iteration.

On the following algorithm steps, the speed and particles positions components are changed according the formula (in coordinate form):

$$v'_{ij} = wv_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (g_j - x_{ij}), \qquad (6)$$

$$x'_{ij} = x_{ij} + v'_{ij}, \qquad (7)$$

where $i = \overline{1,s},\ j = \overline{1,d},$; v'_i and x'_i are new speed and position respectively of particle i; p_i is the best previous position of the particle i; g is the best solution found with whole swarm; w is the *inertial coefficient*; c_1 is the *cognitive coefficient*; c_2 is the *social coefficient*; r_1, r_2 are random numbers uniformly generated on [0;1] interval; they are different for every coordinate.

If during optimisation a particle gets out of search region its respective speed component is set to zero and the particle returns to the nearest border.

The inertial coefficient w means the influence the previous speed of the particle on its new value. The cognitive coefficient c_1 describes the degree of the particle individual behaviour and its intention to get back to the best solution found with it. The social coefficient value c_2 sets the degree of collective behaviour and the intention to move to the particle neighbours' best solution.

Vectors $v_p = c_1 r_1 (p - x)$ та $v_g = c_2 r_2 (g - x)$ define the cognitive and social components of the particle new speed. The random numbers r_1, r_2 add randomness to the

search. Fig.1 shows geometrical interpretation of swarm particles movement direction correction rule (6)-(7).

The coefficient values are selected within ranges [3]:

$$\begin{cases} 0 < w \le 1, \\ 1 < c_1, c_2 \le 2. \end{cases} \quad (8)$$

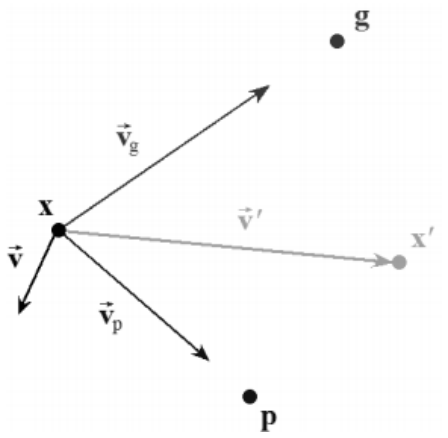

Fig. 1. Geometrical interpretation of rule (6)-(7).

For keeping balance between local and global search the coefficients c_1 and c_2 values are usually equal. Stagnation analysis in [4] has shown that coefficients values $w = 1/2\ln 2 \approx 0.719$ and $c_1 = c_2 = 0.5 + \ln 2 \approx 1.193$ provide in most cases the best results and search stability. Also, there are different ways of setting the swarm parameters dynamically, but adaptation needs some additional initial algorithm iterations: it can lead to objective function (necessary for optimum search) evaluation number increase.

The swarm remembers the best solutions found by particles in separate and by the whole swarm. On initialization the initial particles positions are considered as the best. On every following algorithm iteration after using formulas (6)-(7) the individual best position of every particle p_i and the whole swarm best solution g are corrected according to the rule:

$$\begin{cases} p_i = x_i, \text{ if } f(x_i) < f(p_i), \\ g = p_i, \text{ if } f(p_i) < f(g). \end{cases} \quad (9)$$

So, PSO algorithm scheme is the following:

1. Set algorithm parameters and initialize the swarm: $X = \{x_1, x_2, ..., x_s\}$.
2. Find the best individual value p_i for every swarm particle i, $i = \overline{1,s}$, and the best value for the whole swarm g.
3. Find new positions x_i', $i = \overline{1,s}$, for all swarm particles according to the formula (6)-(7).
4. Check the iteration loop condition. If the condition is *true*, go to item 5, else go to item 2.
5. Output the results.

The typical for optimisation population algorithms conditions are used to determine the iteration loop finish, specifically, reaching maximum iterations number, reaching acceptable solution precision, iteration process stagnation.

3.2 Adaptive particle swarm optimisation

In PSO algorithm there are some free parameters (w, c_1, c_2, s) which can have different optimal value for different problems. Choosing algorithm parameters free values vector is called its adaptation strategy. An adaptive PSO which decreases the probability of premature convergence is Inertia-adaptive PSO Algorithm (IA-PSO) offered in [8]. In this algorithm the inertial coefficient individual value is calculated for every swarm particle by formula:

$$w_i = w_0(1 - dist_i / \max\_dist) \quad (10)$$

where $w_i = rand(0.5, 1)$; $dist_i$ is current Euklid distance from particle i the best solution of swarm g found by the swarm in current iteration, max_dist is the biggest distance from a particle to the best solution found by the swarm in current iteration, e.g. $\max\_dist = \arg\max_{i=\overline{1,s}}(dist_i)$.

Because of the inertial coefficient correction offered the gravity between swarm and particle increases on significant distance between particle and global swarm solution g. This is because in this case w parameter goes down and the particle stops go further from g. To avoid premature convergence, it is necessary to be sure that the particles have enough mobility on late optimisation stages. For reaching this goal the formula of the particles position correction (7) is modified according to dependency $x_{ij}' = (1-\rho)x_{ij} + v_{ij}'$, where ρ is a random value with uniform distribution on [-0.25, 0.25] interval.

3.3 Hybrid PSO algorithm with local search with Nelder-Mead method

Most of swarm intelligence algorithms allow to research search region D but are less effective on researching its small areas. In particle swarm optimisation the local search quality can be improved using lower values of w coefficient. But it decreases the probability of global solution localization and causes premature algorithm convergence. That is why it is better to make local search using special methods (gradient descent method, conjugated gradients method, Newton and quasi-Newton methods) [9]. One of the most popular and effective local search methods which doesn't demand objective function derivatives evaluation is Nelder-Mead algorithm [10]. It is also called deformed polyhedron method.

Hybrid particle swarm optimisation process can be divided into two stages.

1. Optimisation with particle swarm algorithm and global optimum approximate value locating.
2. Search for more exact solution with local search method. In this case local optimisation method gets values found during particle swarm optimisation and

continues locating optimum for the best result found with PSO after several iterations.

3.4 Particle swarm optimisation for discrete problems

In integer optimisation and combinatorial optimisation, the global solution which belongs to some discrete set of possible values should be found. First, the particle swarm optimisation algorithm was developed for solving continuous optimisation problems but using it for solving integer and binary optimisation problems is possible.

In the simplest case the integer programming problems can be solved with using continuous PSO with rounding the result to the nearest integer [11]. So, the solution is searched in D continuous region and before the evaluation of the corresponding objective function value the solution coordinates are recalculated by formula $x_i = [x_i + 0.5]$, where $[x]$ is x number integer part extraction operation. This variant of PSO is called *Discrete Particle Swarm Optimisation* (DPSO).

In [12] BinaryParticle Swarm Optimisation (BPSO) is offered.

3.5 Using PSO for solving mathematical programming problems

Swarm algorithms are intended for solving global optimisation problems on hypercube. For conditional linear and nonlinear optimisation in particular mathematical programming of kind:

$$f(x) \to \min,\ x \in X, \qquad (11)$$

where

$$X = \left\{ x \in D \subset R^d \,\middle|\, g_i(x) \le 0,\ i = \overline{1,m},\ g_i(x) = 0,\ i = \overline{m+1,s} \right\} \qquad (12)$$

the external penalty functions method with *power* or *non-smooth penalty function* (on user choice) is used. Then the problem (11)-(12) in external penalty functions method comes down to solving the problems sequence of kind:

$$F_k(x, r_k) = f(x) + R(x, r_k) \to \min,\ x \in D \subset R^d, \qquad (13)$$

where $k = 1, 2, ...;$ r_k is penalty coefficient which corresponds the condition $r_k > 0$, $\lim\limits_{k \to \infty} r_k = +\infty$, $R(x, r_k)$ is one of external penalty functions.

It is known (in, for instance, [9]) that if points $x^{(k)}$ are the points of global minimum of $F(x, r_k)$, $k = 1, 2, ...,$ then the sequence $\{x^{(k)}\}$ converges to global minimum point x^* of $f(x)$ function on X set i.e. (11)-(12) problem solution.

In practice the problem (13) is solved with fixed penalty parameter value r_k, which is a rather big positive value. The (11)-(12) problem solution will be obtained with some accuracy. In PSO Service parameter r_k has default value of 1000 but it can be changed in parameters input window for increasing problem (11), (12) solution accuracy.

4 PSO Service web resource development tools

PSO Service has web interface. The server side is developed using *Java* programming language and *Spring* framework, the client side is developed using *Thymeleaf* web patterns, *Bootstrap* for decoration. *Hibernate* object-relational mapping system is used for working with database. Table 1 includes information of the tools main characteristics.

Table 1. Overview of technologies for PSO Service web resource implementation.

Technology	Logo	Description
Java		Object-oriented multi-platform programming language
Spring		Universal open-source framework for Java platform
Thymeleaf		XML, XHTML and HTML5 patterns engine for Java
Hibernate		Object-relational mapping framework for Java
Bootstrap		Tools set for setting web sites and web applications view

5 PSO Service main function

PSO Service web resource is bilingual (*English* and *Ukrainian*). It allows the user to learn the theory on swarm algorithms principles, solving optimisation problems with some of these algorithms.

On the main page (Fig. 2) user can read the information about swarm intelligence optimisation methods (*Algorithms*), use algorithms implemented in the service for solving optimisation problems (*Solve*), get information on working with the service (*Help*), get developers contacts (*Contact Us*), authorize (register) in the system (*Login*).

Having learnt about the algorithms user can move to solving the optimisation problem but first he needs to login or register if the user is not registered yet ("*Login*" function).

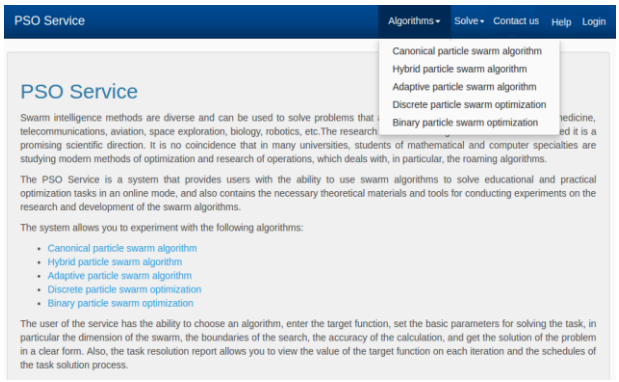

Fig. 2. PSO Service start page.

After login user can start solving optimisation problems with selected algorithm: *Canonical PSO*, *Hybrid PSO*, *Adaptiv PSO*, Discrete *PSO*, *Binary PSO*. If user wants to optimise objective function which is not among test functions he must fill the form on Fig. 3, specifically, the function field using calculator-like buttons and the fields "*Error*", "*Dimension of the task*", "*The size of the swarm*", "*Search space*".

Fig. 3. User function input form.

Also, user can use some of predefined functions on "*Test functions*" tab (Fig. 4).

For making numeric experiments user is to set swarm optimisation parameters on "*Parameter*" tab (Fig. 5). User can change inertial, cognitive, social coefficients, set stopping criterion, number of repetitions etc. User can set, clear, or set default parameter values.

Having made the necessary settings and having sent the form to the server user gets the problem solution. The results page has three tabs: "*Calculation*", "*Input parameters*", and "*Graph/Protocol*".

The "Calculation" tab contains the algorithm execution main results (Fig. 6).

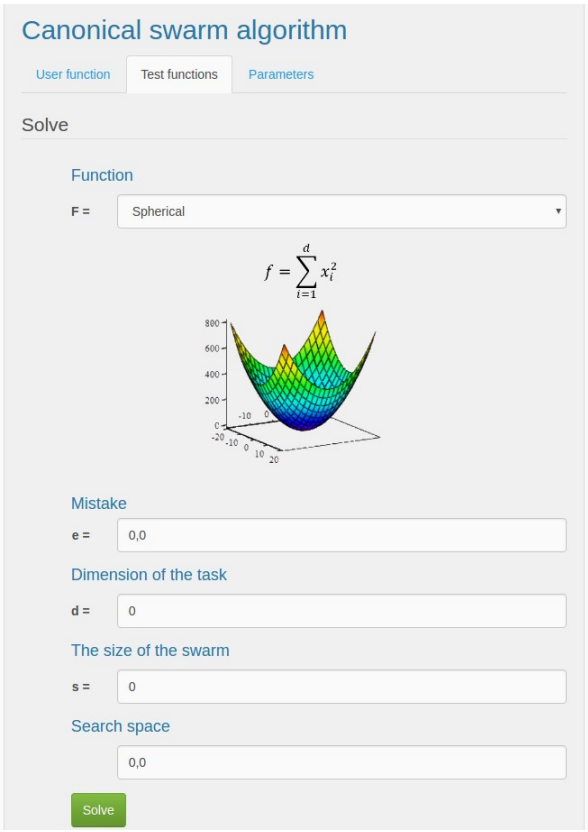

Fig. 4. Test function selection form

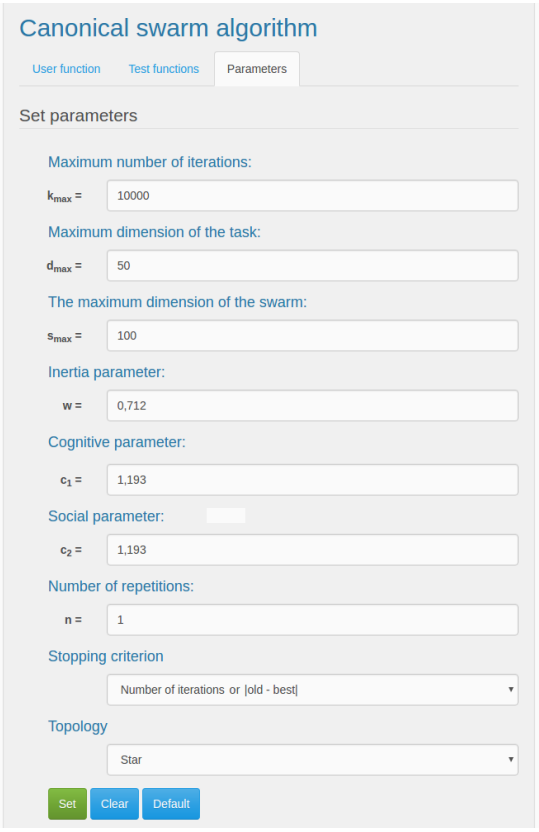

Fig. 5. Particle swarm optimisation parameters setting form.

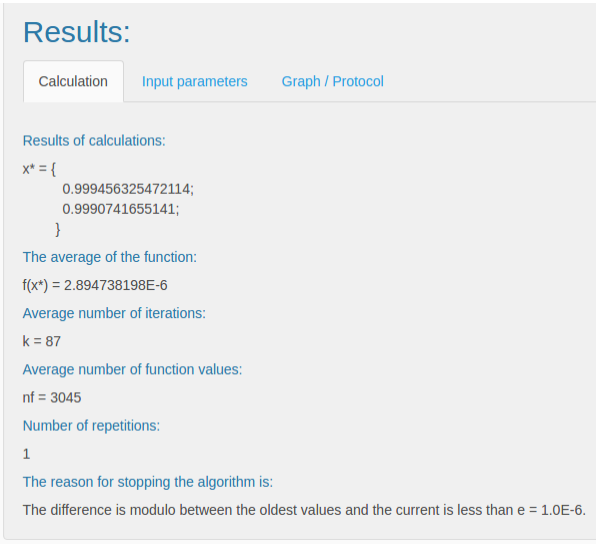

Fig. 6. "Calculation" tab for Rosenbrock function.

"Input parameters" tab includes main parameters of the problem and algorithm (Fig. 7).

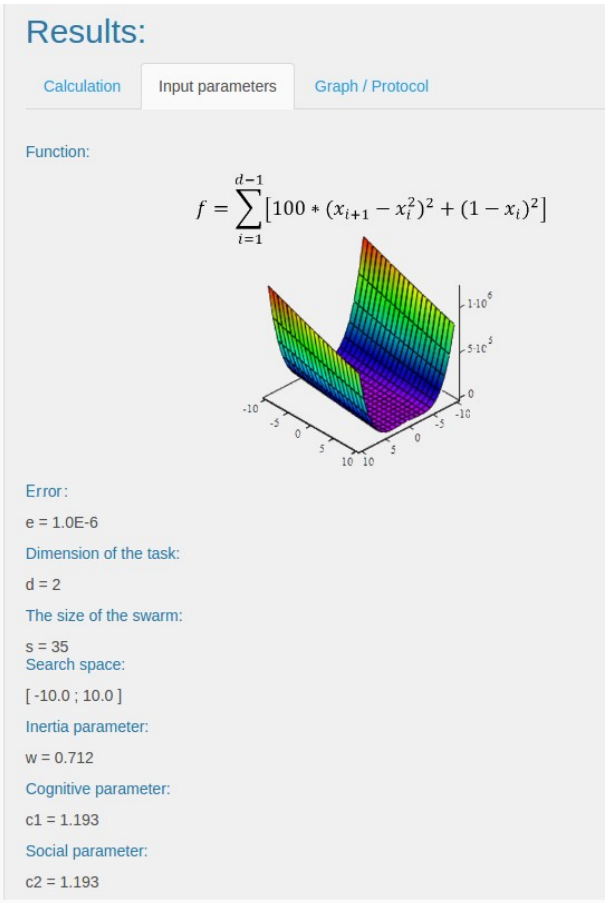

Fig. 7. "Input parameters" tab for Rosenbrock function.

"Graph/Protocol" tab (Fig. 8) contains the dynamic visualisation of optimisation process with the selected algorithm when problem dimension equals 2 and objective function value change on every iteration diagram. Also, problem solution protocol in .xls format is available.

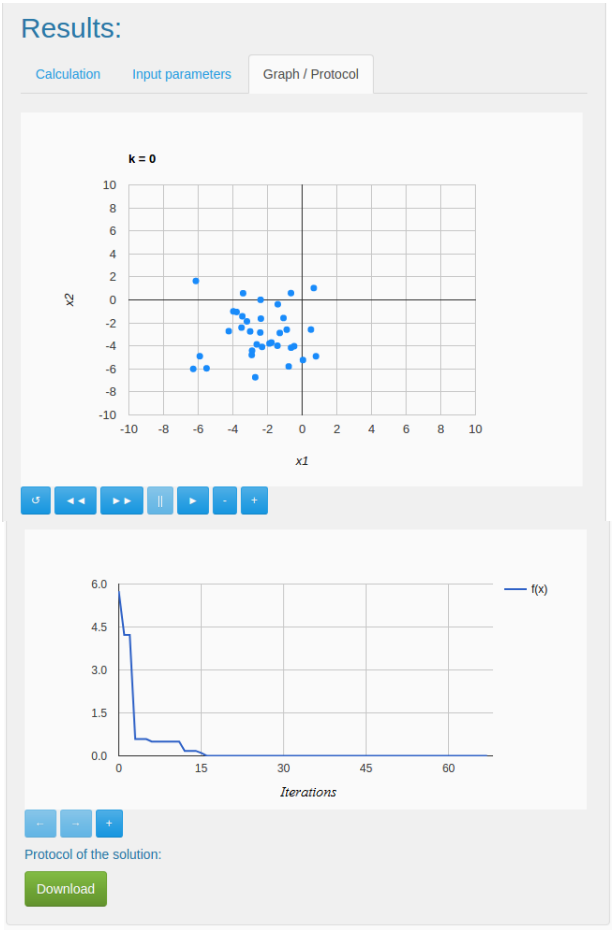

Fig. 8. "Graph/Protocol" tab.

6 Numeric experiment

The set of test functions (Table 2) is used for testing the algorithms implemented in web service accuracy. They are non-linear functions used for optimisation algorithms characteristics assessment, specifically accuracy, convergence etc.

All these functions except Schwefel and Rosenbrock have global minimum in point $x^* = (0, 0, ..., 0)$ with zero value in it.

Non-separable Rosenbrock function (f_4) is one of the most complex. Most of classical optimisation algorithms cannot find its global optimum $x^* = (1, 1, ..., 1)$.

Schwefel function (f_8) global optimum is geometrically remote from the best local minimum. That is why the search algorithm is inclined to go to wrong direction. The global solution for this function is in point $x_i^* \approx 420.9687$, $i = \overline{1,d}$, what is near search region border.

Griewank function (f_6) is a typical separable multi-modal objective function which has many local optimums. Because of topology like that the search algorithms incline to get stuck in one of local optimums.

For making the condition equivalent for every optimisation algorithm being tested the same parameters values were used on every algorithm run.

Table 2. Test functions.

Function name	Formula	*D*	$f(x^*)$
Spherical	$f_1 = \sum_{i=1}^{d} x_i^2$	[-20; 20]^d	0
Elliptic	$f_2 = \sum_{i=1}^{d} (10^6)^{\frac{i-1}{d-1}} x_i^2$	[-20; 20]^d	0
Schwefel 1.2	$f_3 = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$	[-10; 10]^d	0
Rosenbrock	$f_4 = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	[-10; 10]^d	0
Rastrigin	$f_5 = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5; 5]^d	0
Griewank	$f_6 = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	[-100; 100]^d	0
Alpine	$f_7 = \sum_{i=1}^{d} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	[-10; 10]^d	0
Schwefel	$f_8 = -\frac{1}{d} \sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i \rvert} \right) + 418.983$	[-500; 500]^d	0
Ackley	$f_9 = -20 \cdot \exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - \exp\left(\frac{1}{d} \sum_{i=1}^{d} \cos(2\pi x_i) \right) + 20 + e$	[-30; 30]^d	0
Weierstrass	$f_{10} = \frac{1}{d} \sum_{i=1}^{d} \sum_{k=0}^{20} 0.5^k \cos(2\pi 3^k \times \times (x_i + 0.5)) - \sum_{k=0}^{20} 0.5^k \cos(\pi 3^k)$	[-0.5; 0.5]^d	0

In Table 3 the results of using PSO Service for Table 2 test functions solution using canonical PSO with "star" type connections between the swarm particles are given, $\bar{f}_p$ is the average of the function minimal values evaluated on every of *n* algorithm repetitions, $\bar{k}_p$ is iterations number average on every of *n* algorithm repetitions, $f_{\min}$ is function minimum value received after *n* algorithm repetitions, $x_{\min}$ is the vector for which the function has $f_{\min}$ value, $\| x_{\min} - x^* \|$ is Euklid distance between points $x_{\min}$ and x^*.

On testing the condition $| g_{old} - g_{best} | < \varepsilon$ was used as PSO algorithm stop criteria. In the formula g_{old} is the previous best value of objective function for the whole swarm, g_{best} is new best value of objective function for the whole swarm, ε is calculation error.

The following algorithm parameter values for test functions were used:

f_1, f_2, f_3: d=20, s=40, w=0.8, c_1=1.193, c_2=1.193, ε=1e-10, n=20;
f_4: d=20, s=40, w=0.8, c_1=1.193, c_2=1.193, ε=1e-8, n=10;
f_5: d=10, s=40, w=0.9, c_1=1.193, c_2=1.193, ε=1e-20, n=10;
f_6: d=5, s=40, w=0.8, c_1=2, c_2=2, ε=1e-10, n=10;
f_7: d=2, s=20, w=0.9, c_1=1.193, c_2=1.193, ε=1e-15, n=10;
f_8: d=5, s=30, w=0.8, c_1=2, c_2=2, ε=1e-15, n=10;
f_9: d=20, s=40, w=0.8, c_1=1.193, c_2=1.193, ε=1e-15, n=10;
f_{10}: d=10, s=40, w=0.8, c_1=1.193, c_2=1.193, ε=1e-10, n=10.

Table 3. Canonocal SPO testing results.

Function name	$\bar{f}_p$	$\bar{k}_p$	$f_{\min}$	$\| x_{\min} - x^* \|$
Spherical f_1	1.239993E-8	336	1.64199E-09	4.05215E-05
Elliptic f_2	1.8203008E-8	454	1.62316E-09	3.83073E-05
Schwefel 1.2 f_3	4.4579857E-7	1235	4,2467E-08	3.67142 E-4
Rosenbrock f_4	1.0211102335	3892	0.001712	0.05409512
Rastrigin f_5	0.0994959057	9093	2.66E-13	3.6716E-08
Griewank f_6	0.0187167926	9620	3.53E-09	0.000181
Alpine f_7	2.7E-14	1974	9.11E-16	6.44406E-15
Schwefel f_8	52.355709479	5774	0.000113	0.000105469
Ackley f_9	0.6810145400	759	3.24185E-14	3.67643E-14
Weierstrass f_{10}	0.1090308553	462	1.0353E-09	5.93228E-12

Numeric experiment on test functions result has shown that canonical PSO realized in PSO Service is efficient. Also, there was successful efficiency check for other algorithms realized in PSO Service.

7 Conclusion

1. Particle swarm optimisation is widely used in machine learning, specifically for neural networks learning and pattern recognition, parametrical and structural optimisation (shape, size, and topology) in design, biochemistry, biomechanics etc. Considering its effectiveness, it can compete with other global optimisation methods and low algorithmic complexity contribute to its implementation ease.
2. The most promising research directions in this field are theoretical research of particle swarm optimisation convergence reasons and related topics in swarm intelligence and chaos theory areas, combining different algorithm modifications for solving complex tasks, research of SPO in multi-agent computer systems, and research on possibilities of including more complex natural mechanism analogues in it.
3. In this research, the creation of web service for optimisation problems solving with swarm intelligence methods is based. Also, the main requirements for the system like that are set and information technology for its creation is described. The description of the main algorithms and methods implemented in the service and the results of numeric experiment for checking them on test functions are given.
4. In the future, new features are planned to be implemented in PSO Service software: other collective intelligence algorithms; more convenient conditional optimisation functional limitations input; increase methods number of moving it to global optimisation problem on hypercube; extend variants of PSO hybridisation with other local extremum search methods; add new embedded test functions.

5. Created PSO Service can be used: for teaching students, who study swarm intelligence methods; by scientists, who search for different optimisation problem types solving methods effectiveness; by users who use optimisation methods for decision making on real extreme problems solving.

References

1. J. Kennedy, R. Eberhart, *Proceedings of IEEE International conference on Neural Networks*, 1942 (1995)
2. J. Kennedy, R.C. Eberhart, *IEEE Conference on Systems*, 4104 (1997)
3. R. Eberhart, Y. Shi, *Proceedings of the 2000 International Congress on Evolutionary Computation*, 84 (2000)
4. M. Clerc, R. Eberhart, IEEE Trans. Evolut. Computation, **6**, 58 (2002).
5. M. Clerc, *Particle Swarm Optimisation* (USA: ISTE Ltd, 2006)
6. R. Poli, J. Kennedy, T. Blackwell, Swarm Intell., **1**, 33 (2007)
7. Particle Swarm Optimisation Algorithm, https://www.mathworks.com/help/gads/particle-swarm-optimisation-algorithm.html, date of relevance: July 12, 2017
8. M.I. Zhaldak, Y.V. Tryus, *The basics of optimisation theory and methods* (Cherkasy, Brama-Ukraine, 2005)
9. K. Suresh, S. Ghosh, D. Kundu, A. Sen [at el.], Intell. Syst. Design and Applicat., **2**, 253 (2008)
10. J.A. Nelder, R. Mead, Computer J., **7**, 308 (1965)
11. V.Y. Galchenko, A.N. Yakimov, *Particle swarm optimisation population methaeuristic algorithms* (Cherkasy, Tretyakov A.N., 2015)
12. J. Kennedy, R.C. Eberhart, *IEEE Conference on Systems, Man and Cybernetics*, **5**, 4104 (1997)