Particle Swarm Optimization Algorithm

Leiden Institute of Advanced Computer Science
Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Athanasios Agrafiotis, Kevin Noordover

December 2018

1 Introduction

Algorithms that generate a population can be separated into three different classes: evolution-based, physics-based and swarm-based methods.
Evolution-based methods are based on evolution in nature. The population has evolved over generations. The first phase of evolution is called *recombination* which is based on the combination of the best individuals to form the next generation of individuals. This process allow the next generations to optimized over the years together with *mutation*, one of the most popular evolution-inspired technique is Genetic Algorithms(GA) which is based on the Darwinian theory. Other algorithms which are based on the same theory are Evolution Strategy(ES), Genetic Programming (GP) and Biogeography-Based Optimization (BBO). A second algorithm we will experiment with is a swarm-based algorithm. The main characteristic of swarm-based algorithms is that they mimic the social behavior of group of animals.One of the most popular techniques is the Particle Swarm Optimization(PSO) and the Ant Colony Optimization (ACO).
To sum up all these algorithms they work with same mechanism. The frame-

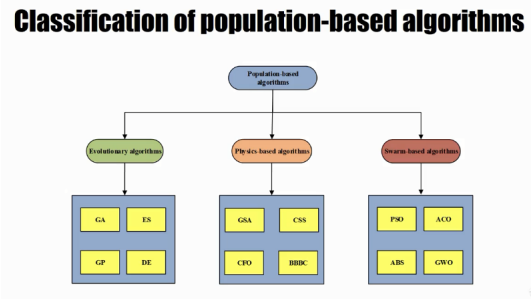


Figure 1: Category of algorithms

work of all these algorithms is almost identical. They start with a population

of random solutions. With the specific operators to perform exploration, exploitation, then estimate the global optimum for a given optimization problem. The variety of the algorithms is based in the field of itemization called No Free Lunch(NFL) [5].
The NFL proves that there is no optimization algorithm to solve all optimization problems. This theorem based on that all algorithms performs equally when averaged across all possible optimization problems. One algorithms can be effective on solving one set of problems and not effective on a different set of problems.
In addition we have the metaphor based algorithms [4]. The problem is that in order to evaluate if an algorithm is good or bad one needs years of research. A lot of researchers simply choose the easiest way to evaluate an algorithm by a crazy factor. After the publication of the Ant Colony Optimization, Particle Swarm Optimization and Genetic Evolution all the next algorithms are painted as Metaphor-Inspired.

2 History of the Algorithm

Reeves in 1983, recommended particle systems to model objects which change shapes and cannot represented of polygons. Examples of those objects is the fire, the smoke and the water. In those objects the *particles* are independent agents and they use a set of rules in order to move. Later in 1987, Reynolds used a particle system to simulate the behaviour of a flock of birds. In a similar way of simulation, Heppner and Grenander(1990) included a roosting place which was attractive to the simulated birds. Those two models played initial part that were later used in the original particle swarm optimization problem. Particle swarm optimization algorithm was finally introduced by Kenned and Eberhart in 1995 It's roots are based in the simulation of social behaviors using tools and ideas taken from computer graphics and social psychology research.
One application of particle swarm optimization algorithm was in neural network training process. The back propagation is one of the most popular methods that has been used for training neural networks. But there are several drawbacks in the use of this algorithm, the training convergence speed is very slow and get stuck in a local minimum. The PSO algorithm has introduced in an attempt to resolve these drawbacks, the results show that this approach can give better accuracy and improve our model behaviour [2].
Another application which particle swarm optimization algorithm has been used is on Clustering analysis. Clustering analysis is applied generally to pattern recognition, color quantization and image classification. Swarm optimization algorithm proposed in the arbitrary data set automatically. PSO can search the best solution and avoid the minimum local value [3].

3 State of the art

4 Stochastic optimization Algorithms

To solve optimization problems employ two conflicting phased:
exploration(diversification) and exploitation(intensification). These two phases are essential to find an accurate estimation of the global optimum for any given optimization problem. The exploration and exploitation phases are in conflict. A simple exploratory behavior mostly results in finding very poor solutions since the algorithm never get the chance to improve the accuracy of the solution. In contrast, mere exploitative behavior results in trapping local solution because the algorithm never get the chance to change the solution surlily. A good balance between the exploration and exploitation leads to finding a very accurate estimation of the global optimum for all types of optimization problems. Finding a good balance between the exploration exploitation is difficult and depends on the shape of the landscape. Bellow in Figure 2 there is a demonstrations of how the particles moves using different exploration and exploitation levels.
The variable which represent the exploration and exploitation levels is the inertia weights. As you can observe when the exploration was at the lowest level the inertia weight was equal to 0. The particle swarm optimization seems to converge really fast. This happens because of the low level of exploration so particles moves really fast to the best optimal solution. In contrast when the inertia weights was equal to 1 which represents a high level of exploration the particles moves really slow to optimal solution because they explore the area. One way to solve real problems is to tune the inertia weights to decrease while the particles moves for instance to start with w equal 0.9 and to stop at 0.2. The behavior of the particles is displayed in the middle figure of 2. In this experiment the particles does not converge fast because the best optimal solution is known from the start.

5 Mathematical Notations, Figures and Tables

5.1 Particle Swarm Optimization

The Particle swarm Optimization algorithm mimics the navigation and foraging of a flock of birds or school of fishes. The PSO algorithm follow a certain rules. The first rule: each point needs to record the distance have been done so far. The second rule: before any movement they need to communicate to be able to find out and update the best distance that the entire points have found so far. So according to rules they try to save and keep record the best distance that they have found so far then, they try to find and update the best position that the entire points have found so far. Each point needs to know it's current direction.

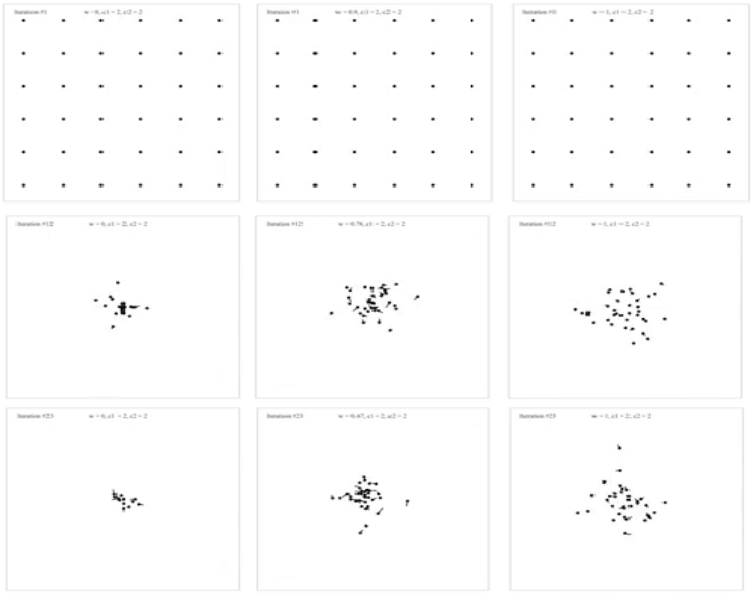


Figure 2: Inertia weights

5.2 PSO search strategy

Table 1: Points

* $\overrightarrow{X_1^t} = [x_1^t, y_1^t]$
* $\overrightarrow{X_2^t} = [x_2^t, y_2^t] \quad => \quad \overrightarrow{X_i^t} = [x_i^t, y_i^t, z_i^d, ...]$
* $\overrightarrow{X_3^t} = [x_3^t, y_3^t]$

In the mathematical equation in table 1. Each point defines its position by the pair of x and y. We can store them in a vector called X. So the X_1^t so the location of the first point in the time t. In the vector we represent the x by i to be able to represent the position of the i-th team member. Also, to be able to move in n-dimensional search space, we can add more variables in the vector X. To update the position vector, we need to define the movement direction and speed. In order to make this done we use a vector called velocity. The V vector is defined by three components: current velocity tendency towards best and tendency towards the and we need to calculate the distance to the personal best(PBEST) and the distance to the GBEST. Each component is multiplied by two times r to randomly increase or decrease it's impact. The next time's

$\overrightarrow{V_i^{t+1}}$	$= 2r_1\overrightarrow{V_i^t}$	$+2r_2(\overrightarrow{P_i^t} - \overrightarrow{X_i^t})$	$2r_3(\overrightarrow{G^t} - \overrightarrow{X_i^t})$
Next Velocity	Current velocity	Distance the Personal Best	Distance the Global Best

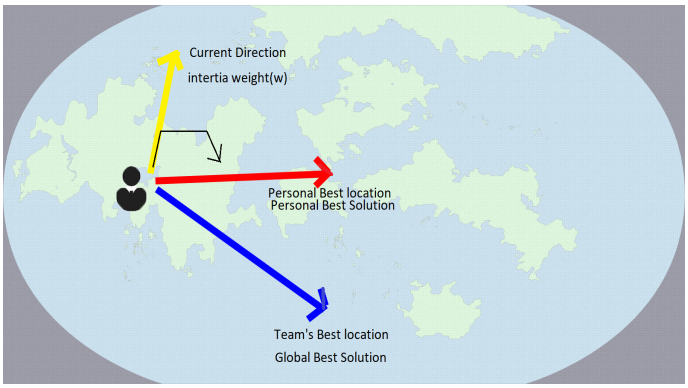


Figure 3: PSO Algorithm Search Strategy

$\overrightarrow{X_i^{t+1}} =$	$\overrightarrow{X_i^t}$	$+\overrightarrow{V_i^{t+1}}$
Position in time t+1	Position in time t	Velocity in time t+1

position is equal to current time position plus the velocity of the next time. We use the velocity of the current time to calculate the position in the next time. Every solution of a given problem is considered as a particle, which is able to move in a search landscape. In order to update the position of each particle, two vectors are considered: the position vector and the velocity vector. These two vectors are updated in each iteration with these two equations. There are slight changes compared to the mathematical model in the analogy.
$\overrightarrow{V_i^{t+1}} = \overrightarrow{wV_i^t} + c_1 r_1(\overrightarrow{P_i^t} - \overrightarrow{X_t^i}) + c_2 r_2(\overrightarrow{G_t} - \overrightarrow{X_i^t})$
Firstly, instead of d we have t, which stands for iteration. Secondly the current velocity is multiplied by a variable called inertia (w).And finally, we have c1 and c2 for the last two components in the velocity vector. There are also different terms for each components of the velocity vector. The first term is called inertia since maintain the current velocity. It maintains the current direction (the movement direction). The second component is called cognitive component or individual component. This is because each particle considers the distance between the PBEST and the current location. The last component, however, is called social component because the particles calculate the distance between its current position found by the entire swarm. In addition the variable G in the social component, there is no subscript because we have just one best solution in the swarm of all particles. The impact of cognitive and social components on the movement of particles can be changed by tuning the coefficients: c1 and c2. The inertia weight $\overrightarrow{wV_i^t}$ is the parameter which tunes exploration and exploitation. However, the coefficient can tune the exploration and the exploitation. As it demonstrate in 4 changing the variables of the coefficients $c1 = 0$ and the parameters of $c2 = 2$ the particles converge to global optimum, which basically represents that the parameter of exploration is equal to 0. On the other hand by changing the coefficients of $c1 = 2$ and $c2 = 0$ the particles never converge

to a global solution and explore the area which means that the exploration is at the highest level and the exploitation is at the lowest level.

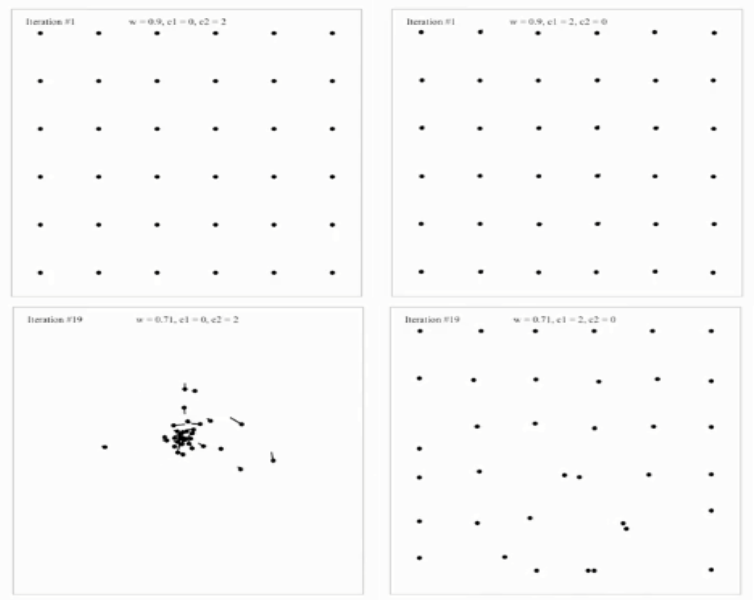


Figure 4: coefficients

6 Pseudo code of PSO

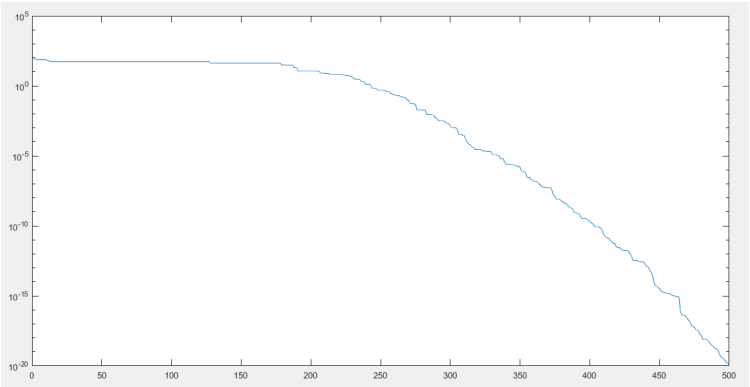
Initialize the controlling parametersN, c1, c2, Wmin, Wmax, Vmax, and MaxIter
Initialize the population of N particles
do
    for each particle
        calculate the objective of the particle
        Update PBEST if required
        Update GBEST if required
    end for
    Update the inertia weight
    for each particle
        Update the velocity (V)
        Update the position (X)
    end for
While the end condition is not satisfied
Return GBEST as the best estimation of the global optimum

Algorithm 1: Particle Swarm Optimization

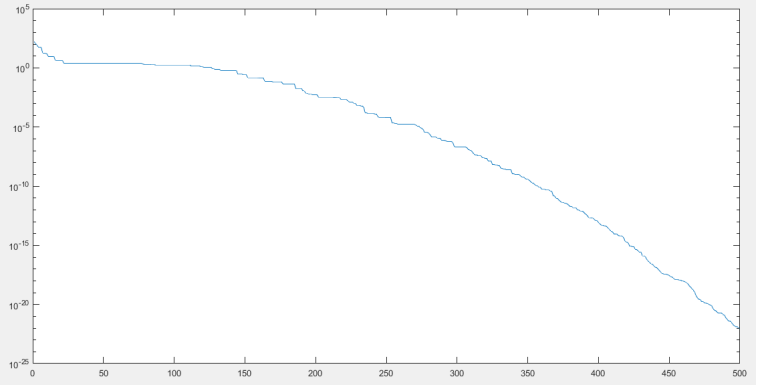
The algorithm starts initializing the controlling parameters. Then initialize the population of the particles, and then there is a main loop. In the main loop first calculate the objective for each particle to be able to update the PBEST and GBEST. After calculating the objective, PBEST and GBEST, we can then update the velocity and position vectors for each particle. And at the end of this loop, we can return GBEST as the best approximation for the global optimum.

7 Algorithm Improvements

The Particle swarm optimization algorithm as we observed so far has gave us the global optimum value(GBEST). In 5(a) we can observe the performance of the algorithm and the global optimum that we have obtained is close to 10^{-20}, the results are good enough already. However there is an improvement that can give us even better result and the algorithm can obtain even better Global optimum solution.
By controlling the velocity vectors towards to an upper pound and lower bound can improve our algorithm's performance [1]. The upper bound and the lower bound prevent particles to go outside of our search area(landscape) by limiting the velocity vectors. The performance of our algorithm displayed in 5(b) the new global optimum value is close to 10^{-25}. The number of iterations is 500 and comparing the results in 5(a) and 5(b) the algorithm performance has improved.



(a) Sample1



(b) Sample2

Figure 5: sample

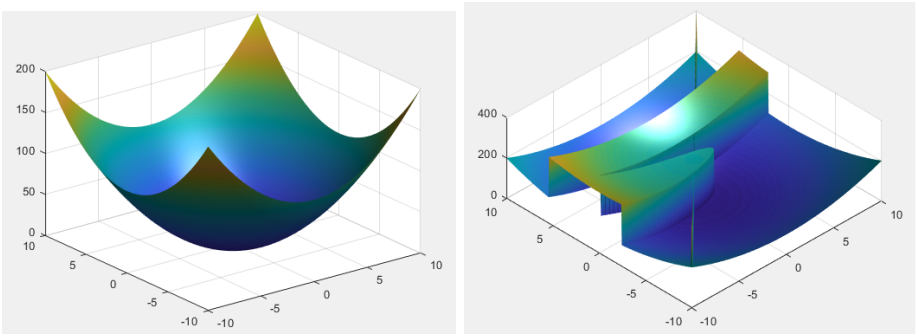
8 Experiments

8.1 1st Experiment with Particle Swarm Optimization Algorithm

The first experiment that was conducted after the implementation of the particle swarm optimization algorithm. In the experiment the efficiency of the particle swarm optimization algorithm was tested using an empty landscape and a landscape with constrains. If the solution is on the constraint area the solution is not feasible. In picture 6 an empty landscape is displayed and in the next picture the same landscape with constraints. Using two variables and 500 iteration the results of the experiment display in table 2. The global optimum variable in an empty landscape is different than the global optimum of the landscape with constraints. In addition the convergence of the algorithm is displayed in figure 7 the convergence of the empty landscape is faster and PSO seems to converge smoothly in contrast with the picture next which the convergence of landscape with constraints seems to move slower because of the constraints in the landscape.

Coordinates	emptyLandscape	LandscapeConstraints
x	1.0e-35*0.5297	0.2470
y	1.0e-35*-.2414	0.9362

Table 2: results

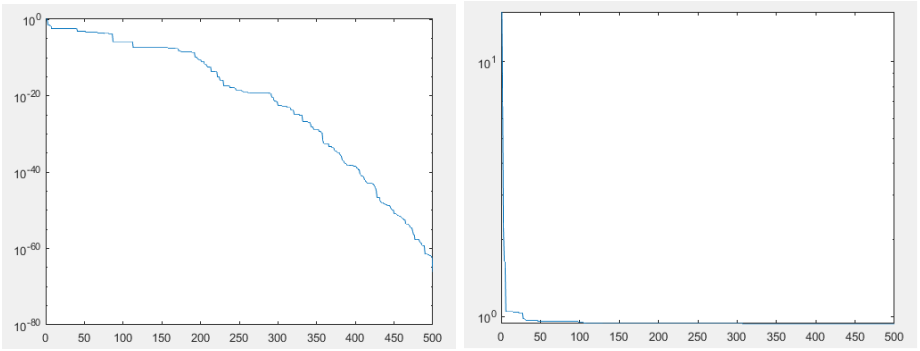


(a) EmptyLandscape (b) Landscape with Constraints

Figure 6: Matlab landscape

8.2 Particle Swarm Optimization Algorithm with different number of particles

The second experiment which conducted using the particle swarm optimization algorithm. Is to solve a sphere objective function with different number of



(a) Convergence of empty landscape (b) Convergence of landscape with constraints

Figure 7: Convergence

particles. Each of the particles represents a solution and in each iteration the position of the particles update their position based on the tunable parameters of exploration and exploitation. By changing the number of the particles seems to affect the result.

Result per Run	Number of Particles	function
5.104e-07	30	sphere
3.7366e-08	30	sphere
7.1477e-07	30	sphere
5.2384e-08	30	sphere
1.6909e-07	30	sphere

Result per Run	Number of Particles	function
4.874e-07	20	sphere
6.4319e-06	20	sphere
8.4327e-07	20	sphere
3.1717e-06	20	sphere
3.7506e-07	20	sphere

Result per Run	Number of Particles	function
0.0003642	10	sphere
0.00048305	10	sphere
0.00025627	10	sphere
1.5169e-05	10	sphere
7.0011e-06	10	sphere

as we can observe through the tables while the number of particles increasing the results getting smaller. Evidently the number of particles seems to affect the

Result per Run	Number of Particles	function
0.21163	5	sphere
0.0024015	5	sphere
0.088442	5	sphere
1.4754	5	sphere
0.0011497	5	sphere

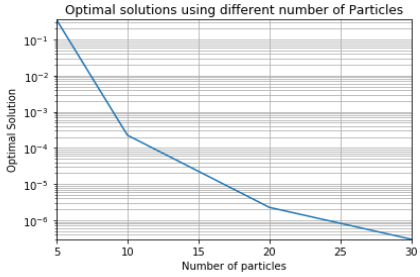


Figure 8: optimal solutions

results because the landscape that particles is getting bigger. In the last picture 8 it is demonstrated how the number of particles effect the result using the same number of iterations and different number of particles. For the purpose of this research the average values used which displayed on the tables of this section.

8.3 3rd Experiment with Particle Swarm Optimization Algorithm

The third experiment which was conducted with particle swarm optimization algorithms was to solve an ackley function. An ackley function is n-dimensional function with a large number of local minima but only one global mininmum. In figure 9 the landscape of an ackley function is displayed based on the matlab. In figure 10 it is demonstrated how does the algorithm converge until it find the optimal solution which is zero. The coeficients c1 and c2 are equal to 2 and the inertia weight decreases from 0.9 to 0.2. As we can observe even in 60 iterations it converges to best solution with 5 number of particles.

9 Conclusion

In this research, we have tested particle swarm optimization for two different functions. In the first experiment, our variant of the particle swarm optimization is tested on a sphere function. First the normal variant and next on a landscape with constraints. In the second experiment, the number of particles was changed for each run so the ideal number of particles could be determined. In the third experiment, we try to solve the ackley function with our implementation.

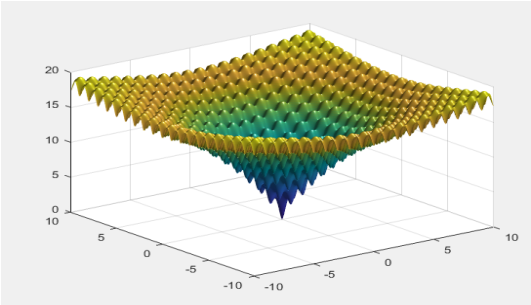


Figure 9: ackley function

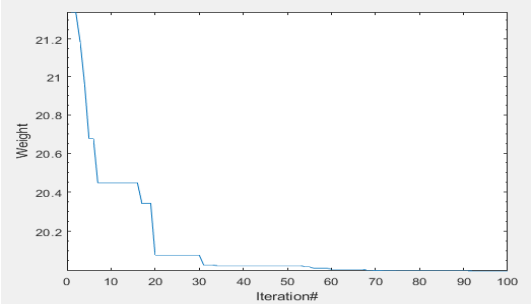


Figure 10: ackley function

On a normal sphere function, the particle swarm optimization algorithm had no problems to find the optimal location. Due to the nature of the sphere function, the initial optimal value was already very close to the optimal coordinate and all other particles were very soon attracted to this location. The convergence curve of this run drops to 0 from the 200th generation, and continues to improve even after the current optimal value is already very close to 0. The sphere function with constraints was a small challenge for the algorithm, since it is no longer enough to just roll down to the lowest point around a particle. The initial best value is a lot larger this time, but the algorithm drops below a best of 1 very quickly. After this, the algorithm needs more time to converge but did find a feasible solution.

In the second experiment, a different number of particles is tested on the sphere function. As expected, a small number performs worse than a larger number. We notice that the found solution improves by a lot when we raise the number from 5 to 10, but that doubling to 20 improves less. 30 particles gives an even better solution, but at this point the number is small enough to be rounded down to 0 and because the solution seems to be converging at this point, a larger number of particles is not needed.

In the final experiment we test the algorithm on an ackley function. The running of the algorithm showed that the swarm could get stuck at a local

optimum for a few generations, but that it would always find a way to improve. This process goes on and on until the global optimum is found. Because this happened in each run, we can conclude that the swarm based optimization algorithm is a reliable way to determine the global optimum in functions like the sphere and ackley functions.

References

[1] Julio Barrera and Carlos A. Coello Coello. "Limiting the Velocity in Particle Swarm Optimization Using a Geometric Series". In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. GECCO '09. Montreal, Québec, Canada: ACM, 2009, pp. 1739–1740. ISBN: 978-1-60558-325-9. DOI: 10.1145/1569901.1570135. URL: http://doi.acm.org/10.1145/1569901.1570135.

[2] Jianxia Chang and Xiaoyuan Xu. "Applying neural network with particle swarm optimization for energy requirement prediction". In: *2008 7th World Congress on Intelligent Control and Automation*. June 2008, pp. 6161–6163. DOI: 10.1109/WCICA.2008.4594562.

[3] Ching-Yi Chen and Fun Ye. "Particle swarm optimization algorithm and its application to clustering analysis". In: *IEEE International Conference on Networking, Sensing and Control, 2004*. Vol. 2. Mar. 2004, 789–794 Vol.2. DOI: 10.1109/ICNSC.2004.1297047.

[4] Kenneth Sörensen. "Metaheuristics – the metaphor exposed". In: In Press (Jan. 2013).

[5] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), pp. 67–82. ISSN: 1089-778X. DOI: 10.1109/4235.585893.