SWARM INTELLIGENCE ALGORITHMS FOR THE PROBLEM OF THE OPTIMAL PLACEMENT AND OPERATION CONTROL OF REACTIVE POWER SOURCES INTO POWER GRIDS

V. MANUSOV[1], P. MATRENIN[1] & S. KOKIN[2]
[1]Novosibirsk State Technical University, Russia.
[2]Ural Federal University, Russia.

ABSTRACT

Deep reactive power compensation allows for reduction of active power losses in transmission lines of power supply systems. The efficiency of the compensation depends on the allocation of reactive power compensation units (RPCUs) at the nodes of a network. In general, investigations devoted to the study of optimal allocation of the compensation units have revealed that it is a static and deterministic optimization problem that can be solved by heuristic methods. However, in real systems, it is reasonable to consider such optimization problems, taking into account the dynamic and stochastic properties of the problems. These properties are the result of equipment failures and operational changes in technical systems. In addition, optimizing the allocation of the compensation units is the NP-hard multifactor problem. Under these circumstances, it is advisable to use the swarm intelligence algorithms. Swarm intelligence is a relatively new approach to solving the optimization problem, which takes inspiration from the behaviour of ants, birds, and other animals. Advantages of swarm algorithms are most evident if problems involve the dynamic or stochastic nature of the objective function and constraints. Contrary to a number of similar studies, this research considers the problem of the optimal allocation of compensation units as a dynamic problem, taking into account the possible random failures of the compensation equipment. The optimization problem has been solved by two Swarm Intelligence algorithms (the Particle Swarm optimization and the Artificial Bee Colony optimization) and Genetic algorithms. It has been aimed at comparing the effectiveness of the algorithms for solving such problems. It was found that swarm algorithms could be successfully applied in the operation control of compensation units in real-time.

Keywords: deep compensation, dynamic optimization problems, operation control, power supply systems, swarm intelligence.

1 INTRODUCTION

This paper is devoted to the problem of decreasing energy losses in power-supply systems up to 10 kV by installing reactive power compensation units (RPCUs). The problem of the loss reduction is approached from different points of view:

- Adopting up-to-date energy-saving fabrication systems and energy-saving electrical products;
- Designing smart grids and equipment complexes (overhead power lines, transformers, switches, etc.) using new principles, methods of transmission and process control;
- Reducing the consumption of reactive energy from generators, arranging reactive power sources at consumers (deep compensation).

ISSN: 1755-7437 (paper format), ISSN: 1755-7445 (online), http://www.witpress.com/journals
DOI: 10.2495/DNE-V12-N1-101-112

In addition, improving the energy efficiency may help to save the environment. For example, calculations made for the situation in Germany show that setting RPCUs reduced power losses by approximately 5.5 billion kWh and prevented emission of about 3 million tons of CO_2 per year [1].

It has been proved that in the radial power supply network of enterprises, implementation of long-term deep reactive power compensation is always profitable and pays off within 1–4 years, because of single-time costs of reactive power units, and increase in power savings in proportion to the operation time of the system. Therefore, this paper does not consider the placement of RPCUs in the network nodes; it considers the problem of the operational control of RPCU powers to ensure minimum active power losses while maintaining the stability of the system. In networks, loads may change dynamically and equipment failures may occur and, therefore, it is necessary that RPCUs should be controlled in real time.

For optimization problems with dynamically changing conditions, it is necessary to apply algorithms that should be first capable of self-organization: that is, automatically adapting to the task and, secondly, should provide efficient solutions fast enough to work in real-time. The most known algorithms possessing these properties are the Genetic Algorithm (GA), Tabu Search, Simulated Annealing, Particle Swarm Optimization (PSO), and Ant Colony Optimization algorithms. The GA [2–5] and the Swarm Intelligence algorithms [5, 6] are used successfully for the optimal placement of RPCUs in power supply systems.

In this study, a comparison of the GA, the PSO algorithm and the Artificial Bee Colony Optimization (ABCO) algorithm in the case of unpredictable time varying conditions of the problem is made. At this stage of the research, the easiest option for the experiments associated with failures of RPCUs has been selected. The purpose of this stage is to analyse the dynamic properties of these algorithms. We aim at creating modifications of these algorithms, wherein our analysis is based on their further use under conditions when, apart from equipment failures, the power distribution of electricity in the grid is varied according to certain distribution functions.

2 PROBLEM DESCRIPTION

2.1 The power supply system considered

In this paper, the power supply system of a uranium enrichment and uranium hexafluoride production plant in the town of Angarsk is considered. The power supply system represents a 0.4 kV substation comprising four sections, and the arrangement of each section is radial. For the power distribution among consumers, power distribution points fed from the substation section bus bars are applied. The most congested 10 distribution points are considered in each section (distribution cabinets), which provide power for asynchronous motors of the ventilation system, electric motors of pumps (the power of each is about 150 kW), and the main supply lines of boost compressors. Each main supply line provides from 10 to 15 compressors with 9 kW induction motors that are supplied by a stub.

Active power losses in transmission lines of this network are high due to the network having a lot of branches and large distances between the nodes. This is attributed to the technological features of the plant considered. The total length of a section transmission line is about 5 km. The total power supply system includes eight sections.

As the deep reactive power compensation is selected, it is suggested to install reactive power sources in such nodes:

- Close to the distribution cabinets;
- Close to the motor control cabinets for the pumps;
- Close to the control cabinets of the first compressor in the main supply line for the compressors.

All in all, each section has 14 nodes for the potential placement of RPCUs.

2.2 Mathematical model

To create a mathematical model of the optimization problem, it is necessary to define, first, an optimization criterion; second, dependent variables; third, constraints; and fourth, functional relationships between these elements. The main aim of the optimization is to minimize active power losses. The dependent variables are the values of the RCPU power in the nodes of a grid. The first constraint is the limitation on tg(ϕ), its value must not be higher than 0.35 [7] and not lesser than 0.1 as a too low value can lead to system instability. The second constraint is related to the powers of an RPCU since the power of each RPCU cannot exceed the reactive power consumption of the corresponding unit. Based on this, the optimization problem is defined as follows:

$$W(Q_{RPCU}) = \Delta P(Q_{RPCU}) \rightarrow min \tag{1}$$

$$Q_{RPCU} = \{Q_1, Q_2, \ldots, Q_n\}$$

$$0 < Q_i < Q_{max\,i}, \; i = 1,\ldots n$$

$$0{,}1 < tg(\phi) < 0{,}35$$

$\Delta P(Q_{RPCU})$ is the total loss of active power within a network using powers of the RPCU defined by Q_{RPCU};

Q_{RPCU} is the RPCU power vector;

Q_i is the RPCU power in the i-th junction (if $Q_{\max\,i} = 0$, then an RPCU is not installed in the i-th junction);

n is the number of junctions where RPCUs are installed; it is shown that there are 14 such junctions for the section considered;

$Q_{\max\,i}$ is the maximum allowable RPCU power level in the i-th junction.

3 METHOD APPLIED

3.1 Population-based algorithms

Stochastic optimizing methods are especially productive for optimization problems that have features such as non-linearity, non-differentiability and a high computational complexity, and are stochastic and dynamic. The methods using principles of nature such as Evolutionary Computation, Swarm Intelligence or Simulated annealing demonstrate the best performance among other stochastic optimization methods. Evolutionary and swarm methods are classified as the so-called population-based methods since they use systems of agents (populations). The term ‘agent’ means a point in the decision space, that is some decision of the optimization problem. The optimization process may be evolutionary or swarming:

- The evolutionary process is based on creating new populations at every new step with regard to the experience obtained by the previous populations (the natural selection).
- Swarming means movements of the agents in the decision space, using a number of rules and an indirect exchange of data between the agents through shared memory. In contrast to evolutionary algorithms, the agents are not created and destroyed and the swarm population has no centralized control system (an ant colony, a bird flock).

The main feature of the population-based algorithms is their self-organization. It ensures the algorithms' ability to explore the decision space automatically regardless of its dimension and topology. This, in turn, provides the flexibility and possibility to find relatively quickly optimal solutions that are almost close to the global ones. Descriptions of the PSO and ABCO algorithms and GA are given in the following text. The detailed descriptions of the natural fundamentals of these algorithms are omitted because they can be easilyfound in other publications of the authors.

In order to make the application and description of the algorithms easier, it is posited that the search space in algorithms is limited between 0.0 and 1.0 for each axis, and for the calculation of the objective function (1) each coordinate x_i is multiplied by the appropriate coefficient (see 3.6).

3.2 Particle swarm optimization

The Particle Swarm Optimization algorithm was first proposed by J. Kennedy and R. Eberhart in 1995 [8]. The algorithm is based on principles of the behaviour of a flock of birds. A flock acts a coordinated group, and every bird acts according to simple rules. It watches the other birds and coordinates its own movement with theirs.

According to the scheme of the swarm algorithms' description [9], the PSO algorithm may be represented by a tuple $\{S, M, A, P, I, O\}$.

1. A set of agents (particles) $S = \{s_1, s_2,\ldots,s_{|S|}\}$. At the j-th iteration the i-th particle is characterized by the state $s_{ij} = \{X_{ij}, V_{ij}, X^{best}{}_{ij}\}$, where $X_{ij} = \{x^1_{ij}, x^2{}_{ij},\ldots, x^l_{ij}\}$ is the position, $V_{ij} = \{v^1_{ij}, v^2{}_{ij},\ldots, v^l_{ij}\}$ is the velocity, $X^{best}{}_{ij} = \{b^1{}_{ij}, b^2{}_{ij},\ldots, b^l{}_{ij}\}$ is the best fitness-functions of the particle position among all the positions it took during the algorithm operation from the 1st to the j-th iterations, and l is the number of dependent variables.
2. Vector $M = X^{best}{}_j$ is the best $X^{best}{}_{ij}$ derived among all the particles.
3. Algorithm A describes the mechanism of the PSO algorithm.
   - The generation of initial positions and velocities ($j = 1$):

$$X_{i1} = rand(0,1),\ i = 1,\ \ldots,\ |S|,$$

$$V_{i1} = rand(0,\ v_{max}),\ i = 1,\ldots|S|,$$

$$X^{best}{}_{i1} = X_{ij},\ i = 1,\ldots|S|,$$

     where $rand(0, 1)$ is the vector of random numbers uniformly distributed from 0 to 1.
   - The calculation of fitness-functions and determination of the best position $f(X)$.

$$X^{best}ij = X_{ij} \mid f(X^{best}{}_{ij}) < f(X_{ij}),\ i = 1,\ \ldots,\ |S|,$$

$$M = X_{ij} \mid f(M) < f(X_{ij}),\ i = 1,\ \ldots,\ |S|),$$

- Particles' movements

$$V_{ij+1} = V_{ij}\omega + \alpha_1(X^{best}_{ij} - X_{ij})rand(0,1) + \alpha_1(M - X_{ij})rand(0,1)\ (i = 1, \ldots, |S|). \quad (2)$$

$$V_{ij+1} = V_{max} |\ V_{ij+1} > v_{max}\ , i = 1, \ldots, |S|,$$

$$V_{ij+1} = -V_{max} |\ V_{ij+1} < -v_{max}\ , i = 1, \ldots, |S|.$$

$$X_{ij+1} = X_{ij} + V_{ij+1}\ , i = 1, \ldots, |S|,$$

$$X_{ij+1} = 1 |\ X_{ij+1} > 1\ , i = 1, \ldots, |S|,$$

$$X_{ij+1} = 0 |\ X_{ij+1} < 0\ , i = 1, \ldots, |S|$$

If a stop-condition is not satisfied, then transit to 3.2.

4. Parameters $P = \{\alpha_1, \alpha_2, \omega, v_{max}\}$. In this paper, the meaning of each parameter is not described as such descriptions have been provided many times in the literature, for example, see [8, 10].
5. Identifiers *I* and *O* are the input and output of the PSO algorithm for interacting with an optimization problem.

3.3 Artificial bee colony optimization

The ABCO algorithm was researched and developed by a number of authors in 2005 [11]. It is based on the simulation of the behaviour of bees in their search for nectar and the indirect exchange of information between them.

According to the description scheme of swarm algorithms [9], the ABCO algorithm may be represented by a tuple $\{S, M, A, P, I, O\}$.

1. A set of agents (bees) $S = \{s_1, s_2, \ldots, s_{|S|}\}$ and $s_{ij} = X_{ij} = \{x^1_{ij}, x^2_{ij}, \ldots, x^l_{ij}\}$.
2. A means of indirect exchange *M* is a list of the best and perspective positions found in the *j*-th iteration, $M = \{N^b_{ij}, N^g_{kj}\}$, $i = 1, \ldots, n^b$, $k = 1, \ldots, n^g$.
3. Algorithm *A* describes the mechanisms of the ABCO algorithm.
   - Initialization of initial positions ($j = 1$) is fulfilled only for a subset of agents termed scouts:

$$X_{i1} = rand(0,1),\ i = 1, \ldots, n^s,$$

   where n^s is the number of scout bees.
   - The calculation of fitness-functions $f(X)$ for scouts at the first step and for all the agents at the next steps.
   - Bee's movements. Among all the agents n^b, the agents with the best values $f(X)$ are chosen, then, out of the rest of the set, n^g agents with the best values are chosen. Using these two sets, the lists of the best and perspective positions *M* are generated. Herewith, the distance between any two positions in *M* over each coordinate must not be less than *rx*. Then, c^b of bees are sent to the vicinity of each best position and c^g are sent to the vicinity of each perspective position:

$$X_{(i-1)\text{cb}+\text{k j}} = N^b_{ij-1} + rand(-1, 1)\cdot rad,\ i = 1,\ldots, n^b,\ k = 1,\ldots, c^b,$$
$$X_{nb\cdot\text{cb}+(\text{i}-1)\text{cg}+\text{k j}} = N^b_{\text{ij}-1} + rand(-1, 1)\cdot rad,\ i = 1,\ldots, n^g,\ k = 1,\ldots, c^g,$$

Finally, scout bees are sent:

$$X_{nb\cdot \mathrm{cb+ng\cdot cg + ij}} = rand(-1, 1)\cdot rad,\ i = 1,\ldots,\ n^s.$$

If the stop-condition is not satisfied, then transition is made to 3.2.

4. Parameters $P = \{n^s, n^b, n^g, c^b, c^g, rad, rx\}$.
5. Identifiers I and O are the input and output as well as I and O of the PSO algorithm.

3.4 Genetic algorithms

The GA started to be used for solving optimization problems during 1960–1970 as a result of investigations made by Ingo Rechenberg and John Holland [12]. The algorithm is based on modelling the mechanisms of natural evolution, such as inheritance, mutation, selection, and crossover. The agents are called individuals and chromosomes. The GA does not use an indirect communication between the agents; instead, the new population of agents S is created again at each iteration. Since the GA is much more widely known than the Swarm Intelligence algorithms, it is inappropriate to give the algorithm's description and, therefore, it is omitted. It is only necessary to specify that we use the classical algorithm with one-point crossover of two parents.

3.5 Selection of parameters

The PSO and ACBO algorithms, as well as other Swarm Intelligence algorithms, are featured with the purpose of setting their heuristic parameters P (coefficients) for a class of problems to be solved for obtaining higher-quality solutions. It is confirmed by a number of researches [8, 13] and the No Free Lunch theorem [14]. In other words, the swarm algorithms cannot always work well, and the parameters require tuning for each type of the problem. The most effective method of setting parameters is a meta-optimization technique [10, 12]. Meta-optimization assumes considering the parameters selection task as a separate optimization task. Herewith, some optimization algorithm resolves the application task, and the meta-optimization selects the parameters of this algorithm for obtaining the best solutions for the application task. In the research [10], the evolutionary adaptation technique was applied for the adaptation of the algorithms with respect to the conditions of the optimization tasks considered. This adaptation method is subject to over-tuning and, therefore, it allows us to obtain good results not only for tasks where the parameters selection was executed but also for other similar tasks. In this paper, we used the parameters found earlier [6] for a similar optimization problem for the PSO and ABCO algorithms. Values of the parameters of the GA equal the values of the parameters found in [5].

3.6 Application of the swarm algorithms to the problem considered

For the interaction of the algorithms with the model of the optimization problem discussed, it is necessary to set up a correspondence between position X of an agent and the vector specifying the RPCU arrangement in the grid (Q_{RPCU}). In our present paper [6], vector X is used not as an RPCU power vector, but as the coefficients' vector, so the power of each RPCU was determined as the product of the X vector element and the calculated maximum allowable power of the RPCU in the corresponding junction: $Q_i = x_i \cdot Q_{imax}$. If for the considered variant

of the RPCU arrangement the value of tg(ϕ) does not fit the limitations, then the value of the optimization criterion is added with the *penalty* value and the *penalty* >> $W(Q_{RPCU})$).

4 EXPERIMENTS

4.1 The purpose of the experiment

The experiment simulated the failure cases of one of the RPCU in the grid. It is necessary to carry out a real-time automatic adjustment of powers of another RPCU to preserve the stability of the system and minimize the active power losses. There are two alternative ways in the case of using the population-based algorithms for dynamically changing optimization problems.

- As soon as there is a change in the problem's conditions, the optimization algorithms are interrupted and then the algorithm is run again for a new optimization problem. In this case, the dynamically changing problem is seen as a set of separate unrelated static optimization problems. This approach ignores the achieved experience of the population of the algorithm before changing the conditions of the problem. However, it allows the algorithm to get out of a local extremum and start the search process from scratch, that is start from the initialization of a new random population. Let us call this way 'restart' or 'the first way'.
- Never stop the search algorithm. Thus, when the conditions of the problem are changed, the search process does not start from scratch, but from the state in which the population was at the moment of changing the conditions. In this instance, there is a chance to quickly find an effective solution to the modified problem, but this approach increases the risk of getting stuck in a current local extremum. Let us call this way 'without restart' or 'the second way'.

The experiments are aimed at comparing the effectiveness of these two ways applied for the optimization algorithms considered. A priori, it has been suggested that the first way is preferred for the PSO algorithm and GA and the second way is preferred for the ABCO algorithm because only the ABCO algorithm ensures that all the agents of the population will not appear in the vicinity of a single solution.

4.2 Experiment description

During the experiments, each of the algorithms considered (PSO, ABCO, GA) solved the optimization task (1) separately. The parameters of the algorithms are given in Table 1.

The experiments for each algorithm were performed as follows:

Table 1: Algorithm parameters.

Algorithm	Number of agents	Heuristics parameters
PSO	100	$\alpha_1 = 2.03$ $\alpha_2 = 2.32$, $\omega = 0.87$, $v_{max} = 0.9$
ABCO	100	$n^s = 10$, $n^b = 15$, $n^g = 10$, $c^b = 4$, $c^g = 3$, $rad = 0.01$, $rx = 0.05$
GA	100	$p_{xover} = 0.9$, $p_{mutation} = 0.2$

1. Find the quasi-optimal solution X^* of task (1) when all the RPCUs are operational.
2. For each $node_i$ of the grid that has the RPCU (i = 1, ..., n).
   - Assume that the $RPCU_i$ refused and ceased to function completely.
   - Continue the process of optimization of the algorithm for the task modified, that is use the restart way described in 4.1. The process is stopped after 200,000 iterations after the conditions of the task have been changed.
   - Start the optimization algorithm from scratch without taking into account the previous solution, that is use the way without restart described in 4.1. The process is stopped after 200,000 iterations after the conditions of the task have been changed.

Table 2: Solutions of the task (1).

Algorithm	Way	id of the RPCU failed	ΔP(100)	ΔP(500)	ΔP(1,000)	ΔP(2,000)	ΔP(20,000)
PSO	restart	1	312.02	311.92	311.92	311.92	311.90
PSO	w/o restart	1	324.48	324.48	324.48	324.48	324.48
ABCO	restart	1	311.99	311.42	310.97	**310.70**	**306.14**
ABCO	w/o restart	1	**311.80**	**311.37**	**310.93**	310.77	310.34
GA	restart	1	346.34	346.34	344.27	333.88	316.42
GA	w/o restart	1	343.28	339.19	339.19	335.69	318.15
PSO	restart	7	**311.32**	311.31	311.31	311.31	311.31
PSO	w/o restart	7	318.06	318.06	318.06	318.06	318.06
ABCO	restart	7	311.54	**311.02**	**310.87**	**310.80**	304.43
ABCO	w/o restart	7	311.36	311.09	310.96	310.68	**300.57**
GA	restart	7	354.69	344.42	344.42	338.26	315.68
GA	w/o restart	7	345.26	345.26	345.26	342.53	319.40
PSO	restart	9	352.67	352.67	352.67	352.67	352.67
PSO	w/o restart	9	392.46	392.46	392.46	392.46	392.46
ABCO	restart	9	324.59	324.09	323.90	323.55	**315.19**
ABCO	w/o restart	9	**324.19**	**323.54**	**323.36**	**323.15**	321.44
GA	restart	9	392.87	392.87	392.87	392.87	368.21
GA	w/o restart	9	408.93	396.98	387.31	387.31	366.19
PSO	restart	12	358.78	358.78	358.78	358.78	358.78
PSO	w/o restart	12	398.97	398.97	398.97	398.97	398.97
ABCO	restart	12	330.12	329.61	329.36	329.03	**318.74**
ABCO	w/o restart	12	**329.83**	**329.06**	**328.83**	**328.63**	321.72
GA	restart	12	409.76	407.65	399.73	392.86	382.14
GA	w/o restart	12	415.28	409.39	408.97	399.58	377.26
PSO	restart	13	311.57	311.57	311.56	311.56	311.56
PSO	w/o restart	13	322.89	322.89	322.89	322.89	322.89
ABCO	restart	13	311.58	311.40	311.33	311.23	**300.08**
ABCO	w/o restart	13	**311.07**	**310.91**	**310.91**	**310.76**	304.81
GA	restart	13	352.56	342.97	340.22	332.53	312.68
GA	w/o restart	13	344.40	344.40	344.40	327.93	319.65

As the solution of the optimization problem is expected in real-time, it is important how quickly a quasi-optimal solution to the task will be found after changing the conditions of the task. Therefore, the results were recorded at 100, 500, 1,000 and 2,000 iterations. The final results obtained after 200,000 iterations are rather of theoretical value.

4.3 Experimental evidences

The experimental results are shown in Tables 2–4. Table 2 lists the results of solving the optimization problem (1) for the cases of failure of various RPCUs (the results are not shown for all the RPCUs). The first column defines the algorithm used, the second one shows the approach used for accounting changing conditions of the problem (see 4.1), the third gives the number of the failed RPCUs in the correspond experiment. The following columns show the values of the optimization criterion (active power losses, kW) after the given number of algorithm iterations. Iterations were counted after changing the conditions of the problem, that is, from step 2.1 of the description in paragraph 4.2.

Table 2 shows that almost in all these cases, the ABCO algorithm provided the best solution, regardless of the number of iterations. The PSO algorithm with restart gave significantly better results than the PSO algorithm without restart. Moreover, the performance of the PSO algorithm without restart is hardly improved by increasing the number of iterations, which confirms that the PSO algorithm is unable to get out of a local extremum in case of changing the conditions of the task. It is attributed to the fact that when the PSO algorithm convergences in the neighbourhood of an extremum, elements (X^{best}_{ij} - X_{ij})

Table 3: Comparison of the algorithms' efficiency, the average deviation.

		average deviation ΔP of the best, %				
Algorithm	Way	100	500	1,000	2,000	200,000
PSO	restart	4.132	4.127	4.125	4.123	4.118
PSO	w/o restart	8.025	8.024	8.024	8.024	8.024
ABCO	restart	2.728	2.652	2.612	2.563	0.597
ABCO	w/o restart	**2.617**	**2.551**	**2.522**	**2.475**	**0.492**
GA	restart	17.057	14.606	13.486	11.323	6.530
GA	w/o restart	13.952	13.064	12.633	11.687	6.897

Table 4: Comparison of the algorithms' efficiency, the max. deviation.

		maximum deviation ΔP of the best, %				
Algorithm	Way	100	500	1,000	2,000	200,000
PSO	restart	12.565	12.565	12.565	12.565	12.565
PSO	w/o restart	25.173	25.173	25.173	25.173	25.173
ABCO	restart	4.072	3.982	3.9372	3.883	2.065
ABCO	w/o restart	**3.852**	**3.795**	**3.784**	**3.735**	**1.981**
GA	restart	28.558	27.896	25.411	24.643	19.893
GA	w/o restart	30.289	28.442	28.309	25.363	18.361

and (M - X_{ij}) in eqn (2) appeared to be very small, and the speeds of the particles are too small for them to leave the extremum in a reasonable time. The GA gave the worst results, and its effectiveness with restart is better for some cases, but for others the way without restart is better. The GA is able to get out of a local extremum by mutation, which resulted in the ability of the agents of the populations to appear in arbitrary points of the search space.

Table 2 lists the detailed results only for some RPCUs. Total results averaged over all the RPCUs are shown in Tables 3 and 4. Table 3 shows the deviation of the values of the optimization criterion (ΔP) from the best value found among all the algorithms after 200,000 iterations. The deviations are averaged over all the RPCUs. The maximum deviation is shown in Table 4, using the same patterns of the table.

The experiments have proved that the PSO algorithm is most sensitive to the manner in whichthe dynamic changes of optimization problems' conditions are accounted. Therefore, the restart of the PSO algorithm should be performed as soon as the conditions of the problem are changed. For the GA, it is slightly more preferable to perform a restart, while on the contrary, for the ABCO algorithm, taking into account earlier solutions helps to effectively solve non-permanent tasks. In general, the ABCO algorithm shows a significantly higher efficiency of the considered problem of the RPCU control. This can be explained by the fact that the agents of the algorithm are always a swarm of bees dispersed over several areas of the solutions' search space and never appear in the vicinity of one extremum. In this case, the ABCO algorithm is more complicated to implement and much more labor-intensive in terms of selecting the heuristic parameter, as the ABCO algorithm has seven algorithm parameters, as shown in Table 1.

5 CONCLUSION

1. The task of managing the sources of reactive power in networks up to 10 kV considered as a dynamic optimization problem is being solved in real-time. Since loads in the grid can vary in real-time, to improve the compensation efficiency, it is advisable to provide operative control of the reactive power sources located in the grid. For the experiments, the failures of RPCUs were simulated, resulting in dynamic changes of the problem's conditions.
2. To solve this optimization problem we applied population-based algorithms: the GA, and PSO, and ABCO algorithms. In the case of solving dynamic optimization problems, these algorithms can either operate continuously in response to changes in the problem or restart every time the conditions change. These two ways can be defined as 'without restart' and 'restart'.
3. The experiments show that the PSO algorithm is necessary to perform the restart; otherwise, the algorithm does not go out of a local extremum, as the agents of the PSO algorithm tend to congregate into the vicinity of one extremum (Tables 2 and 3). The GA provides approximately the same efficiency with restart and without it, and it is obvious that decreasing the probability of the mutation can make the restart more preferable. However, the results of the GA are significantly worse than the results of both Swarm Intelligence algorithms (Tables 3 and 4). The ABCO algorithm shows the best performance and the approach without restart appears to be more effective (Tables 3 and 4). If the parameters of the ABCO algorithm are tuned properly, then the ABCO agents are always scattered across multiple extrema, it allows the algorithm to take into account previously found solutions as well as to find new ones.

4. Further, we plan to make the optimization problem considered in this paper more explicit and bring it closer to real conditions. For this purpose, the model of the power supply system will include not only failures of RPCUs, which, in fact, occur infrequently, but will also take into account the dynamic changes of loads in grids and operating conditions of consumers.

NOMENCLATURE

ABCO	Artificial Bees Colony Optimization
GA	Genetic algorithm
PSO	Particle Swarm Optimization
RPCU	Reactive power compensation unit(s)
tg(ϕ)	the reactive power to active power ratio

REFERENCES

[1] Energieeinsparung durch Blindleistungskompensation, available at: www.zvei.org/Publikationen/Blindleistung.pdf

[2] Da Silva, E.L., Gil, H.A. & Areiza, J.M., Transmission network expansion planning under an improved genetic algorithm. *IEEE Transactions on Power Systems*, **15**(3), pp. 1168–1174, 2000. http://dx.doi.org/10.1109/59.871750

[3] Paterni, P., Vitet, S., Bena, M. & Yokoyama, A., Optimal location of phase shifters in the french network by genetic algorithm. *IEEE Transactions on Power Systems*, **14**(1), pp. 37–42, 1999. http://dx.doi.org/10.1109/59.744481

[4] Mantawy, A.H., Abdel-Magid, Y.L. & Selim, S.Z., Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem. *IEEE Transactions on Power Systems,* **14**(3), pp. 829–836, 1999. http://dx.doi.org/10.1109/59.780892

[5] Manusov, V., Tretyakova, E. & Matrenin, P., Population-based algorithms for optimization of the reactive power distribution and selection of the cable cross-section in the power-supply systems. *Applied Mechanics and Materials,* **792**, pp. 230–236, 2015. http://dx.doi.org/10.4028/www.scientific.net/AMM.792.230

[6] Manusov, V., Kokin, S. & Matrenin, P., Optimal placement of reactive power sources in power supply systems, using particle swarm optimization and artificial bees colony optimization, 2015.

[7] Order of the Russian Federation Ministry of Industry of 22.02.2007 no. 49, Moscow, 2007.

[8] Kennedy, J. & Eberhart, R., Particle swarm optimization. *Proceeding of IEEE International Conference on Neural Networks*, **4**, pp. 1942–1948, 1995. http://dx.doi.org/10.1109/ICNN.1995.488968

[9] Matrenin, P.V. & Sekaev, V.G., Sistemnoe opisanie algoritmov roevogo intellekta [Systems approach to swarm intelligence]. *Programmnaja inzhenerija*, **12**, pp. 39–45, 2013. http://dx.doi.org/10.1109/SIBCON.2015.7147143

[10] Matrenin, P.V. & Sekaev, V.G., Particle swarm optimization with velocity restriction and evolutionary parameters selection for scheduling problem. *Proceeding of the Inter-*

national Siberian Conference Control and Communications (SIBCON), IEEE: Omsk, pp. 1–5, 2015.

[11] Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. & Zaidi, M., The bees algorithm – a novel tool for complex optimisation problems, Manufacturing Engineering Centre, Cardiff University, Cardiff, UK, 2005.

[12] Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press: Ann Arbor, 1975.

[13] Pedersen, M. & Chippereld, A., Simplifying particle swarm optimization. *Applied Soft Computing,* **10**(2), pp. 618–628, 2010. http://dx.doi.org/10.1016/j.asoc.2009.08.029

[14] Wolpert, D.H. & Macready, W.G., No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation,* **1**(1), pp. 67–82, 1997. http://dx.doi.org/10.1109/4235.585893