# Simulated Annealing and Evolutionary Algorithm for Base Station Location Problem: a Comparison of Methods

**Evgenii Skakov, Vladimir Malysh**

*Faculty of Physics, Mathematics and Computer Science, Lipetsk State Pedagogical University, Russian Federation, Lipetsk, wallkirya@mail.ru, vmalysh@mail.ru*

**Contribution to the state of the art**

DOI: **10.7251/JIT1502088S** UDC: **004.7:[621.39:681.586**

**Abstract:** A modifications of the evolutionary algorithm and simulated annealing method for solving the base station location problem for creating a wireless data network is introduced in the article. By the way of computer simulation a comparison of speed and accuracy of solutions obtained by the proposed methods and the method of exhaustive search is produced. The study revealed that new simulated annealing method show better results than the modified evolutionary algorithm.

**Key words:** base station location, evolutionary algorithm, simulated annealing, wireless networks, optimization, SIR.

## Introduction

An important part of designing a wireless data network is the synthesis of its topological structure. It implies planning a territorial distribution of base transceiver stations and connection to the customers.

As it is known [3], [7], [9] the problem of designing a network (including wireless) can be reduced to the problem of finding the minimum of a functional present value when probability and time and structural characteristics of the network are limited and the appurtenance requirement for many variants of the network architecture to the field of technically implemented solutions is met. Detailed classification of location algorithms for base stations (BS) in UMTS 3G networks is given in paper [9].

Among the shortcomings of the works on this problem, we should note the following:

- solution to the problem of BS location using methods that do not show high calculation speed (branch and bound method, a heuristic method of Lagrange, etc.);
- absence of limits considering the level of attenuation of signal during propagation from the BS to the client and vice versa;
- most of the studies do not take into account the level of inter-cell interference;
- most of the models do not involve the use of several types of base stations.

This work aims to create a model that does not have the above drawbacks. In it NP-hard problem is solved by placing the BS using metaheuristic optimization methods, namely evolutionary algorithm (EA) and simulated annealing (SA).

### Purpose of the work

Formulate a mathematical model of the problem of optimal base stations location; suggest a modifications of the evolutionary algorithm and simulated annealing for solving this problem; compare the effectiveness of the proposed algorithms.

### Formulation of the problem

The problem of locating base stations is that we have $N_{tp}$ customers, each of them must be connected to the base station. The base station may be installed at one of the $N_{ps}$ potential sites (candidate places). There are $N_{types}$ types of base stations, different in their characteristics. The problem is reduced to minimizing the total cost of the installed base stations under certain restrictions.

We will present a solution to the problem in the form of a pair of vectors (single dimensional arrays) of integers - $Y$ and $X$ (explanation: here and further recording species *A[j]* means an appeal to the *j*-th element of the vector *A*; array elements are numbered starting with 1):

- $Y$ – a vector containing $N_{ps}$ elements. The elements may take on integer values in the range $[0; N_{types}]$. If $Y[i]=0$, then a base station is not installed at the *i*-th position-candidate. If $Y[i]=w$, then base station *w*-type is set at the *i*-th potential site;
- $X$ – a vector containing $N_{tp}$ elements. The elements may take integer values in the range $[1; N_{ps}]$. If $X[i]=w$, then the *i*-th client is connected to the base station at the *w*-th potential site.

The client can be connected to the position-candidate only if it has the base station installed:

$$Y[X[i]] \neq 0 \qquad \forall i \in \{1, 2, \ldots, N_{tp}\}. \tag{1}$$

Let $b$ – the vector whose elements are required bandwidth of clients; $\beta$ – the vector whose elements are the maximum performances of different types of base stations. For each installed BS the total required traffic must not exceed the highest possible performance of the equipment:

$$\sum_{i=1}^{N_{tp}} p_{is} \cdot b[i] \leq \beta[Y[i]] \quad \forall s \in \{1, 2 \ldots, N_{ps}\}$$

$$\text{where} \tag{2}$$

$$p_{is} = \begin{cases} 1, & \text{if } X[i] = s \\ 0, & \text{else} \end{cases}.$$

Let $P_{BS}^{\max}$ – the vector whose elements are the maximum powers of base stations of various types; $P_{TP}^{\max}$ – the vector whose elements are the maximum powers of customers; $P_{BS}^{tar}$ – the vector whose elements are the sensitivities of the base stations of different types; $P_{TP}^{tar}$ – the vector whose elements are the sensitivities of clients; $G$ – a two-dimensional array (matrix) of dimension $N_{tp} \times N_{ps}$, each element of which $0 \leq G[i][s] \leq 1$ reflects the level of attenuation between the client *i* and the candidate place *s*.

Despite the attenuation of the signal along the path from the BS to the client, the power reaching the transmitter from receiver must exceed the minimum target power:

$$\frac{G[i][X^{GA}[i]] \cdot P_{BS}^{\max}[Y^{GA}[X^{GA}[i]]]}{P_{TP}^{tar}[i]} \geq 1 \qquad \forall i \in \{1, 2, \ldots, N_{tp}\}. \tag{3}$$

Restriction (3) is compiled for the downlink mode. A similar restriction for the uplink mode (despite signal attenuation along the path from the client to the BS, the power reaching from the client to the BS must exceed the minimum target power) is as follows:

$$\frac{G[i][X^{GA}[i]] \cdot P_{TP}^{\max}[i]}{P_{BS}^{tar}[Y^{GA}[X^{GA}[i]]]} \geq 1 \qquad \forall i \in \{1, 2, \ldots, N_{tp}\}. \tag{4}$$

According to sources [7] and [9], the objective function to be minimized must be as follows:

$$F = \sum_{s=1}^{N_{ps}} Cost[Y[s]], \tag{5}$$

where Cost – the vector of prices (including installation) for different base stations types.

However, this idea has an obvious disadvantage. The problem of locating the BS may have multiple admissible solutions with the same value of the total equipment cost. In this case, it is not clear which of them is considered to be the best. In this paper, we propose to calculate an objective function as follows:

$$F = \left( \sum_{s=1}^{N_{ps}} Cost\left[Y[s]\right] + \sum_{i=1}^{N_{tp}} SIR^{i}_{dB} \cdot k \right). \qquad (6)$$

The first summand is the total cost of the complex. The second summand is responsible for keeping the level of SIR for all customers of the system. SIR is the signal-to-interference ratio [5]. In the general case, it is calculated as follows:

$$SIR_{dB} = 10\log_{10}\left( \frac{P_{signal}}{P_{interference}} \right). \qquad (7)$$

According to our problem, we calculate SIR for each client. The received signal from the BS, which is connected to our client, appears in the numerator. Signals from other BS that interfere appear in the denominator:

$$SIR^{i}_{dB} = 10\log_{10} \frac{G[i][X[i]] \cdot P^{\max}_{BS}[Y[X[i]]]}{\sum\limits_{s=1, s \neq X[i]}^{N_{ps}} G[i][s] \cdot P^{\max}_{BS}[Y[s]]}. \qquad (8)$$

*k* – a coefficient with simultaneous consideration two factors in the objective function: the cost of creating the network and the SIR-level of customers. *k* has the dimensions of the complex of base stations cost, thereby keeping the dimension constant in the formula (6), as SIR - a dimensionless quantity. *k* should provide a "penalty" for the low level of SIR and a "reward" for the high one. In this paper, taken *k*=-10 conventional units (*k* is a negative value, because we solve the minimization problem, and therefore, a good high level of SIR is to reduce the objective function).

**Evolutionary algorithm for the problem of base stations location.**

A population considered in the evolutionary algorithm "exists" for a predetermined number of generations $N_{gen}$. The population consists of $N_{pop}$ individuals, each of them in our case corresponds to a solution of the base stations. Each individual solution comprises two chromosomes *Y* and *X*, which have been described above.

The basis of the evolutionary algorithm work is the evaluation operation of an individual's fitness in the population. According to the principle of natural selection, the higher is the fitness level, the greater is the probability of an individual to reach the next generation. In our problem, the level of fitness of an individual is inversely proportional to the value of the objective function that this individual presents. Accordingly, the smaller is the objective function *F*, the higher is the fitness of the individual.

In the evolutionary algorithm, each of its iterations (generation) consists of a selection operation from the initial parent population and breeding operation (includes crossover and mutation). As the best solution the best individual in the last generation of the population concerned returns. Figure 1 (flowchart) gives a general view of the evolutionary algorithm.

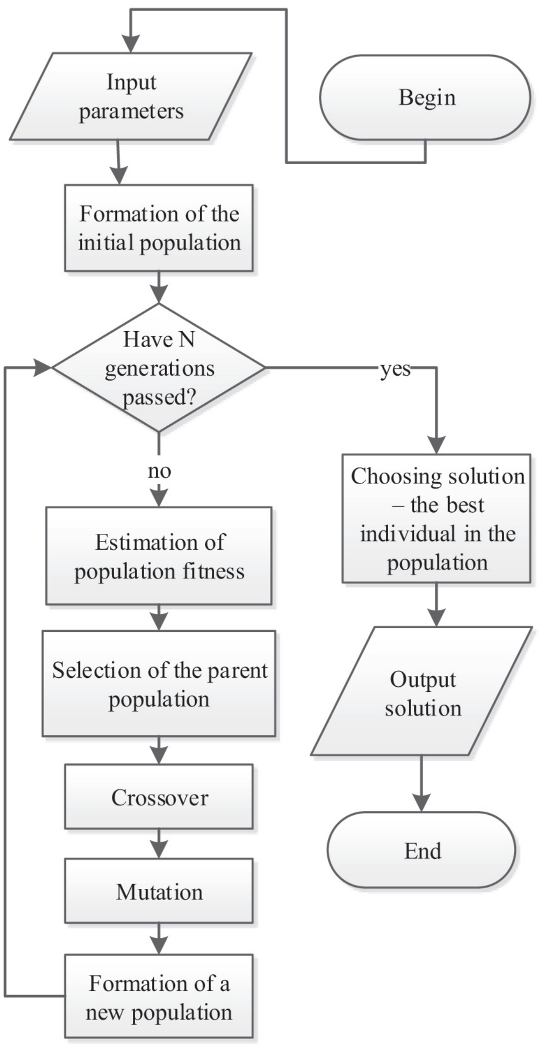
Input parameters
Begin
Formation of the initial population
Have N generations passed?
yes
no
Estimation of population fitness
Choosing solution – the best individual in the population
Selection of the parent population
Output solution
Crossover
End
Mutation
Formation of a new population

**Figure 1.** Flowchart of the evolutionary algorithm

All decisions will be checked for validity, i.e. matching constraints (1), (2), (3) and (4). Only "viable" individuals will be included in the new population, i.e., those that satisfy all constraints.

**Selection of the parent population.**

Selection means to choose (based on fitness of individuals) those chromosomes that will participate in the creation of the next generation. The most common are 3 selection methods:

1) Roulette method. The method for minimizing problems can be briefly described as follows: "If the value of the objective function *i*-th individual *k* times smaller than the objective function value of the *j*-th individual, then probability of inclusion of the *i*-th individual to the parent pool must be *k* times greater than the probability of similar *j*-th individual".

2) Tournament method. It implements a sequence of tournaments to choose *N* individuals. Each tournament is built on a random choice from a population of *k* elements and selecting the best individuals among them. The most common tournament selection with *k*=2.

3) Ranking method. The probability of getting into the parent pool is not proportional to a fitness function value, but is proportional to a spot that is taken by an individual in a population that value-ordered by objective function.

Suppose that we have *N* individuals. Then the rank of the *i*-th individual will be assigned for the following reasons: the worst individual has rank 0, next to the "quality" – 1... the best individual has rank (*N*-1). The probability of finding the *i*-th individual in the parent population is:

$$p_i = \frac{2r_i}{N(N-1)}. \tag{9}$$

The so-called elitist strategy should be mentioned as well. It is about protecting the best individuals in the transition from one generation to another. In the classical evolutionary algorithm the fittest individual of *i*-th generation does not always go to generation (*i*+1). The elitist strategy is used to prevent the loss of the best individual, which is guaranteed to be included in the new population. In this paper we will use the elitist strategy in addition to the selection methods discussed above.

**Reproduction procedure.**

Creating descendants is made using genetic operators. In the evolutionary algorithm two genetic operators are usually used: crossbreeding operator (crossover) and mutation operator.

Each solution to our problem of locating the BS is encoded by two chromosomes, so the procedure will differ from classic crossover.

It can be seen that for each solution vector *X* completely determines nonzero elements in vector *Y*. Vector *Y* only determines what type of BS is set to a specific location.

Since chromosomes *X* and *Y* are dependent on each other, we cannot cross them individually (as there would be many wrong decisions) [7]. Therefore, we will only cross vectors $X_1$ and $X_2$ of parental species.

Crossover operator runs as follows: a pair of individuals is randomly chosen from the parent population. Next, for each pair of individuals selected this way a position of a gene in chromosome *X* is randomly selected, the position, which determines the so-called crossover point. This point is an integer in the interval [1, *L*-1], where *L* – length of chromosome *X* (in our case $L=N_{tp}$). As a result of crossing two parental chromosomes there appears a descendant with $X_{child}$ chromosome which consists of the genes of the first parent in positions from 1 to $l_c$, and it consists of the genes from the other parent in positions from ($l_c$+1) to *L*. Modification of the one-point crossover is an *n*-point (eg, multi-point) crossover. It is similar to the one-point, but crossbreeding is held for *n* points in it.

For the obtained descendant vector $X_{child}$ the mutation operation will be applied. The operation of mutation means that with some small probability one of the elements of the vector changes its value to a random integer from the range [1; $N_{ps}$].

After receiving the vector $X_{child}$ of descendant-individual, we must form its chromosome $Y_{child}$. Chromosome $X_{child}$ uniquely determines which elements of the vector are nonzero. For example, we have 5 potential sites, 4 clients and 3 types of BS (and the smaller the serial number of the type of BS is, the cheaper such stations are). Let $X_1$=[1 5 2 3] and $X_2$=[2 5 5 3]. As a result of crossover we got $X_{child}$=[1 5 5 3] for the child. This means that the child chromosome $Y_{child}$ will have the form $Y_{child}$=[$V$ 0 $V$ 0 $V$] where $V$ - is an integer from the range [1; $N_{types}$]. ’

The non-zero elements of the vector are recommended to be chosen in the following way:

1) if $Y_1[i]=w$ and $Y_2[i]=0$, then $Y_{child}[i]=w$;
2) if $Y_2[i]=w$ and $Y_1[i]=0$, then $Y_{child}[i]=w$;
3) if $Y_1[i]=Y_2[i]=w$, then $Y_{child}[i]=w$;
4) if $Y_1[i]=w$, and $Y_2[i]=z$ ($w\neq z$), then we have several methods of selecting the type of BS, which will be situated in the descendant at the $i$-th place:

a) take the type that has minimal cost (to minimize the objective function);
b) take the type of BS with greater productivity;
c) take any $w$ type, or take randomly $z$ type with 0.5 probability.

**Another form of solution representation.**

To describe the solution of the BS location task for solving by simulated annealing we need to use such a unit of data as a structure. The structure is a composite data type that stores a set of different variables (fields), united by one name. In our case structure contains three fields:

- variable *type*, indicating what type of BS is located in the potential site (if *type* = 0, the BS is not located);
- variable *cl_nbr* – the number of clients connected to this site;
- one-dimensional array (vector) of integers *CL* (dimension cl_nbr), containing the number of customers connected to this place.

We will present the solution as a vector *Sol*, whose elements are the records that correspond to the potential sites. Appeal to a separate variable recording (so-called field) is denoted with the symbol ‘.’. That is *Sol[3].type* means the type of BS third potential site, and, for example, *Sol[4].CL[2]* is the number of the 2-nd customer connected to a BS in a place №4.

The customer can be connected to the candidate place only if it is already installed at the base station. So, for each location, whose *Sol[s].cl_nbr≠0*, we must have *Sol[s].type≠0*.

Then the restrictions (2)–(4) take the form (10)–(12) respectively:

$$\sum_{i=1}^{Sol[s].cl\_nbr} b\big[Sol[s].CL[i]\big] \le \beta\big[Sol[s].type\big] \quad \forall s \in \{1,2,\dots,N_{ps}\} \tag{10}$$

$$\frac{G\big[Sol[s].CL[i]\big]\big[s\big]\cdot P_{BS}^{max}\big[Sol[s].type\big]}{P_{TP}^{tar}\big[Sol[s].CL[i]\big]} \ge 1 \quad \forall s \in \{1,2,\dots,N_{ps}\},\ \forall i \in Sol[s].CL. \tag{11}$$

$$\frac{G\big[Sol[s].CL[i]\big]\big[s\big] P_{TP}^{max}\big[Sol[s].CL[i]\big]}{P_{BS}^{tar}\big[Sol[s].type\big]} \ge 1 \quad \forall s \in \{1,2,\dots,N_{ps}\},\ \forall i \in Sol[s].CL. \tag{12}$$

The objective function (6) takes the form

$$F = \sum_{s=1}^{N_{ps}} Cost\big[Sol[s].type\big] + \sum_{s=1}^{N_{ps}} \sum_{i=1}^{Sol[s].cl\_nbr} SIR_{dB}^{si}\cdot k. \tag{13}$$

We need to sort out the types of base stations according to the ascending price. Then we assume that the $i$-th type of BS is more expensive than ($i$-1)-th and cheaper than ($i$+1)-th.

**Simulated annealing for the BS location problem**

Simulated annealing based on the simulation of physical processes that occur during crystallization of substances from liquid to solid (for example, during the annealing of metals) [1]. The process takes place at a gradually decreasing temperature. The transition of an atom from one cell of the crystal lattice to another occurs with some probability, and the probability decreases with the decreasing of temperature [2].

Kirkpatrick has applied the above ideas for solving the optimization problems [4]. The simulated annealing method belongs to the class of local search algorithms. At each step of the algorithm in the neighborhood of the current solution *x* a solution *w* is chosen. As in the local descent algorithm, if the objective function *f(w)* of the solution *w* will be better (in the case of minimization – less) than *f(x)*, then *w* replaces *x* as the current solution. However, the method of the SA has a feature that helps it to avoid "getting stuck" in the local optima: there is some probability of transition to the solution w, even if *f(w)* is worse than *f(x)*. This probability is given by [6]:

$$P(t) = exp\left((f(w) - f(x))/t\right), \tag{14}$$

where the parameter *t* in analogy with the physical process is known as the temperature. Obviously, the higher the temperature, the greater the probability of transition to a worse solution is. In the course of the algorithm the parameter *t* is decreasing constantly. In this paper we consider exponential annealing, and therefore the temperature at the (*k*+1)-th step can be calculated using the temperature on the *k*-th step with the formula:

$$t_{k+1} = \alpha t_{k}, \tag{15}$$

where the parameter $\alpha \in (0;1)$ is a cooling factor. The algorithm stops working when the temperature becomes close to zero.

Separately, let us talk about choosing the initial temperature $t_0$, as this parameter is very important for the correct operations of the method. Kirkpatrick, the author of the SA method [4] gives only general advice, which means that the value $t_0$ should be chosen very large. However, in [1], the authors derived a formula for the calculation of $t_0$. Let $p_0$ – the objective probability of making the worst decisions in the early steps of the algorithm, $\Delta f_{max}$ – the maximum possible difference between the values of the objective function of two neighboring solutions. Then the formula for the $t_0$ takes the form:

$$t_0 = -(\Delta f_{max}/Ln(p_0)). \tag{16}$$

Obviously, it is very difficult to calculate $\Delta f_{max}$ previously, so we have to use a rough estimate of this value in the calculation. In our task the parameter $\Delta f_{max}$ should be approximately equal to the cost of the most expensive BS.

**Solution neighborhood.**

The concept of "neighborhood" is the most interesting in the method of simulated annealing. It is badly formalized and for each specific optimization problem is distinctive. In the base station location problem we have the solution presented in the form of a number of base stations with customers connected to them. We must decide which solutions are closest to ours. We devised a method of forming a neighborhood solution through implementing small changes in the current solution. The new solution from the neighborhood of the current solution can be obtained by one of six methods (operations) [8]:

1) Change the type of one BS to another BS, which is cheaper.

This kind of operation (*Sol[s].type=Sol[s].type*-1) is possible for each element of the array *Sol*, which has *Sol[s].type*>1. After changing we must check new configuration for compliance with limitations (10)–(12).

2) Change the type of one BS to another BS, which is more expensive.

This operation (*Sol[s].type=Sol[s].type*+1) is possible for each element of the array *Sol*, and which has *Sol[s].type*$\neq N_{types}$ and *Sol[s].type*≠0.

3) Reconnect one client (i.e. the connection to another BS).

This operation is possible for each client. If we are seeking for a new BS for the *i*-th customer, we have to try consistently to connect it to one of the potential sites (except his current location), starting with the neighbor to the customer *i* and checking for compliance with limitations (10)–(12). Obviously, the place *w*, to which we want to connect the customer must have an active BS (*Sol[s].type*≠0).

4) Deletion of one BS.

This operation is enabled for each active potential site. For each of the customers leaving the station *s* we launch the operation 'reconnect one client'. If all the *s*-th station customers can be connected to other BS, deleting means *s*-th station is possible.

5) Addition of one BS. This operation is available for each empty space (*Sol[s].type*=0). We set in place *s* a new station and try to connect it to the closest customer (let his number be *i*). If such a connection is possible (for the *s*-th site run constraints (10)–(12)), then new solution *Sol* is allowable. Customer with number *i*, of course, must be disconnected from the old BS.

6) Relocation of one BS. This operation is enabled for each active potential site. Let us relocate BS from the place *s*. Then we must consistently try to fit it into the empty potential sites, starting with the closest to the site *s*. At the new place *w* constraints (11) and (12) must be performed.

**Pseudo-code for the SA algorithm.**

Below is the general scheme of simulated annealing algorithm for the base station location problem [8].

```
1.  Build Curr (initial potential solution).
    Notations: $t$ – initial temperature; $t\_min$ – temperature is which annealing is terminated; $N\_iter$ – number of iterations at each temperature; Sol – solution from Curr neighborhood; Best – the best solution.
2.  Best=Curr.
3.  Sol=Curr.
4.  Apply to Sol one of the 6 steps described above
    (selecting operation performed randomly).
5.  If Sol – forbidden solution (does not satisfy one
    of the constraints (10)–(12)), then move to step 3.
6.  If $F(Sol)<F(Curr)$, then move to step 8.
7.  Let $w$ – a random number in the range [0; 1];
    $P(t)=\exp((F(Curr) - F(Sol)) / t)$.
    If $w>=P(t)$, then move to step 3.
8.  Curr=Sol.
9.  If at this value of $t$ was done N_iter iterations,
    then decrease $t$.
10. If $F(Curr)<F(Best)$, then Best=Curr.
11. If $t>t\_min$, then move to step 3, otherwise move
    to step 12.
12. Return Best.
```

**Computer simulation**

The developed algorithm was implemented as software in the programming environment Embarcadero Delphi XE5. With this software, some computational experiments on finding the optimal location of the base stations and connection of customers to them were carried out. The simulation was performed on a computer with Intel Core i5-3470 processor and 6GB of RAM.

The first series of computational experiments was devoted to the study of the performance and accuracy of proposed methods by comparing them with the exhaustive search method (ES) for the problems of small dimension. The principle of the exhaustive search algorithm is very simple: we must try all possible solutions of the task, we must remove all the wrong solutions, and choose the best among the remaining (in terms of the objective function). In our problem, the total number of combinations of chromosome pairs *X* and *Y* is equal to $N_{ps}^{Ntp} \,.\, ((N_{types}+1)N_{ps}-1)$. It is obvious that exhaustive search method does not solve the problem of base stations in polynomial time.

We fix the parameters of the EA (according to [7]): ranking method of selection, 2-point crossover, $N_{gen}$=100, $N_{pop}$=100, the elitist strategy was used. We fix the parameters of simulated annealing algorithm (according to [8]): $\alpha$=0.97, $p_0$=0.95, *N_iter*=20. The results of the experiment are shown in Table 1. Each cell in the table contains two lines: the top line is the time of the algorithm's operation in seconds, the bottom line is the value of the objective function in conventional units. Here and later the time of solving the problem and the objective function value are given as the average of 100 runs of the algorithm.

The presented data suggest that for the small dimensional problems both evolutionary algorithm and simulated provide accurate values of the objective function as well as the exhaustive search method, which gives an exact solution for each run of the algorithm. The developed modifications of EA and SA methods allows to find the solution of base station location problem in a reasonable time, many orders of magnitude faster than the exact method of exhaustive search

**Table 1.** Comparison of EA, SA and ES methods on small-size problems

| Size of task ($N_{tp}$×$N_{ps}$× $N_{types}$) | Method | | |
|---|---|---|---|
| | ES | EA | SA |
| 3×5×2 | 0,030 sec<br>27382 | 0,093 sec<br>27382 | 0,015 sec<br>27382 |
| 3×7×2 | 0,710 sec<br>26784 | 0,093 sec<br>26784 | 0,015 sec<br>26784 |
| 3×10×2 | 65,19 sec<br>26516 | 0,109 sec<br>26516 | 0,015 sec<br>26516 |
| 5×5×2 | 0,606 sec<br>26776 | 0,096 sec<br>26776 | 0,015 sec<br>26776 |
| 5×7×2 | 31,59 sec<br>26570 | 0,093 sec<br>26570 | 0,016 sec<br>26570 |
| 5×10×2 | 5140,7 sec<br>26776 | 0,109 sec<br>26776 | 0,017 sec<br>26776 |
| 7×5×2 | 14,65 sec<br>26458 | 0,097 sec<br>26458 | 0,016 sec<br>26458 |
| 7×7×2 | 1390 sec<br>26396 | 0,100 sec<br>26396 | 0,016 sec<br>26396 |
| 7×10×2 | 455798 sec<br>26230 | 0,111 sec<br>26230 | 0,018 sec<br>26230 |

Then, a series of computational experiments to compare the effectiveness of the proposed evolutionary algorithm and simulated annealing method for solving various size problems was carried out. We fix the parameters of the EA (according to [7]): ranking method of selection, 2-point crossover, $N_{pop}$=100, the elitist strategy was used. We fix the parameters of simulated annealing algorithm (according to [8]): $\alpha$=0.97, $p_0$=0.95, $N\_iter$=20. The results are shown in Table 2. Each cell in the table contains two lines: the top line is the time of the algorithm's operation in seconds, the bottom line is the average deviation of the solutions from the best known solution.

Simulation was performed as follows: first, each of the tasks we solved by simulated annealing method. Next, we solve the same problems by the EA, limiting it work time by the time that it took the SA to solve those problems. The presented data suggest that modified simulated annealing method shows better results than the evolutionary algorithm (at a same fixed runtime).

**Table 2.** Comparison of evolutionary algorithm (EA) and simulated annealing (SA)

| Size of task ($N_{tp}$×$N_{ps}$×$N_{types}$) | Method | |
|---|---|---|
| | EA | SA |
| 50×50×3 | 0,247 sec<br>8,293 % | 0,247 sec<br>3,698 % |
| 100×100×3 | 0,695 sec<br>12,177 % | 0,695 sec<br>8,515 % |
| 150×150×3 | 1,415 sec<br>9,329 % | 1,415 sec<br>5,267 % |
| 200×200×3 | 2,539 sec<br>12,050 % | 2,539 sec<br>9,262 % |
| 500×500×3 | 16,83 sec<br>6,602 % | 16,83 sec<br>4,426 % |

## Conclusions

Provided computer simulation allows the following conclusions to be made:

- proposed modifications of the evolutionary algorithm and simulated annealing can solve the problem of base station location;
- EA and SA can find a solution of our task within a reasonable time;
- for the problems of small dimension the EA and SA results coincide with the method of exhaustive search. Moreover evolutionary algorithm and simulated annealing method require much less time than ES;
- SA shows better results than the EA (at a same fixed runtime).

## References

[1] Das, H. et al. (1990). *Scheduling of serial multiproduct batch processes via simulated annealing*. Computers & chemical engineering, 14(12), 1351–1362.

[2] Dreo, J. et al. (2006). *Metaheuristics for Hard Optimization*. Springer.

[3] Ermolaev, S.I. (2010). *Optimal base station location.* Telecommunication Sciences, 1(1), 349–355. (in Russian).

[4] Kirkpatrick, S. et al. (1983). *Optimization by Simulated Annealing*. Science, 4598(220), 671–680.

[5] Koskie, S. and Gajic, Z. (2005). *Nash game algorithm for SIR-based power control in 3G wireless CDMA networks*. IEEE/ACM Transactions on Networking, 13(5), 1017–1026.

[6] Luke, S. (2013). *Essentials of Metaheuristics*. Lulu.

[7] Skakov, E.S. and Malysh, V.N. (2015). *Evolutionary algorithm to solve the task of base station location*. Bulletin of Ryazan State Radio Engineering University (RSREU), 51(1), 46–52. (in Russian).

[8] Skakov, E.S. and Malysh, V.N. (2015). *Simulated annealing for base station location problem*. Control Systems and Information Technology, 60(2), 90–94. (in Russian).

[9] St-Hilaire, M. et al. (2012). *Efficient solution of the 3G network planning problem*. Computers & Industrial Engineering, 63(4), 819–830.

Submitted: May 27, 2015.
Accepted: October 26, 2015.