DISTRIBUTION PROBLEM SOLVED BY ARTIFICIAL INTELLIGENCE METHODS

Abstract: *This paper deals with the use of artificial intelligence that can solved many problems in entrepreneurial and economic field. The artificial intelligence includes genetic algorithms and neural networks. These methods can solve many problems from practice. The distribution problems belong among them. The correct solution of such problems enables us to minimize the cost, save time and nature. The methods mentioned in the article can be used for many similar cases and various places.*

Keywords: *Distribution problems, genetic algorithms, neural network*

1. Introduction

They are problems that are necessary to solve in practice. The distribution problems belong among them. The correct solution of such problems enables us to minimize the cost, save time and nature. The genetic algorithms and neural networks can help us with these problems.

2. Genetic algorithms as an optimization task

The genetic algorithms simulate the evolution of human population. During the calculation by means of genetic algorithms we use such operators as selection, crossover and mutation. The selection means the choice of the best individuals. The crossover represents the exchange of so-called chromosomes among single individuals of the population. The mutation means the modification of a part of a particular chromosome if a random change happens.

Genetic algorithms operate such that the initial population of chromosomes is created first; this population is changed by means of genetic operators until the process is finished. One cycle of the reproduction process is called the epoch of evaluation of a population (generation).

The aim of genetic algorithm as an optimization task is to divide the set of N existing objects into M groups. Each object is characterized by the values of K variables of a K-dimensional vector. The aim is to divide the objects into groups so that the variability inside groups is minimized.

Let $\{\mathbf{x}_i; i = 1,2,\ldots,N\}$ be a set of N objects. Let x_{il} denote the value of l-th variable for i-th object. Let us define for $i = 1,2,\ldots,N$ and $j = 1,2,\ldots,M$ the weights

$$w_{ij} = \begin{cases} 1 & \text{if the i - th object is a part of j - th group ,} \\ 0 & \text{otherwise .} \end{cases}$$

The matrix W = [w$_{ij}$] has the following properties

$$w_{ij} \in \{0;1\} \text{ and } \sum_{j=1}^{M} w_{ij} = 1 .$$

Let centroid of *j*-th group $c_j = [c_{j1}, c_{j2}, \ldots, c_{jK}]$ be calculated in such a way, that each of its elements is a weighted arithmetic mean of relevant values, i.e.

$$c_{jl} = \frac{\sum_{i=1}^{N} w_{ij} x_{il}}{\sum_{i=1}^{N} w_{ij}}.$$

The inner stability of *j*-th group is defined as

$$S^{(j)}(W) = \sum_{i=1}^{N} w_{ij} \sum_{l=1}^{K} (x_{il} - c_{jl})^2$$

and its total inner group variance as

$$S(W) = \sum_{j=1}^{M} S^{(j)} = \sum_{j=1}^{M} \sum_{i=1}^{N} w_{ij} \sum_{l=1}^{K} (x_{il} - c_{jl})^2 .$$

The distances between an object and a centroid can be calculated in this case by means of common Euclidean distances

$$D_E(\mathbf{x}_p, \mathbf{x}_q) = \sqrt{\sum_{l=1}^{K} (x_{pl} - x_{ql})^2} = \left\| \mathbf{x}_p - \mathbf{x}_q \right\| .$$

The aim is to find such matrix $W^* = [w^*_{ij}]$, that minimizes the sum of squares of distances in groups from their centroids (over all *M* centroids), i.e.

$$S\left(W^*\right) = \min_W \{S(W)\}$$

3. Neural network as an optimization task

It is possible to use the neural network with so called unsupervised learning for optimization processes that is based on evaluation of the difference (distance) of the weighted vector $\mathbf{w}$ of the neural network from the vector of input pattern $\mathbf{x}$ and search of neuron, whose weighted coefficient have the minimum distance of $\mathbf{w}$ from $\mathbf{x}$. This neuron, which won among the neurons of the network, has the right to adjust its weights and the weights of neurons in its surroundings and thus the response on submitted learning pattern to better value. After submitting of a further learning pattern it can win another neuron of the net that can adjust its weights and the weights of neurons in its surroundings and thus to increase better answer etc. The groups are thus created in the net that in certain places of the network optimally respond to certain symptoms of submitted patterns, as well as unknown patterns.

The „map" of patterns is created. This network was presented by Kohonen in 1982 and it is called Kohonen self organizing map. The network contains n distributing nodes that are fed by single components of the vector of input pattern $\mathbf{x}_p = \left(x_{1p}, x_{2p}, \ldots, x_{ip}, \ldots, x_{np}\right)^T$ and N efficient neurons distributed on imaginary surface (one-layer network), for example on the surface of specific configuration such as rectangular structure, where the efficient neurons are found in individual crossing lines of the rectangular structure (there are used other structures, such as with glory in efficient neuron using other structure, such as hexagon). The neurons have among themselves the bindings, so that it is possible to define the surroundings of each neuron. Further, it is possible to think over lateral activity of neuron on the neighbouring neurons according to the rule: the nearest surroundings are represented by exciting bindings and further surroundings by inhibitive bindings. The action of the i-th component x_i on the j-th neuron is done by weight coefficient w_{ji} (real value). The weight vector of the j-th neuron is $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jn})^T$.

The input vector (for the p-th input pattern) has the form

$$\mathbf{x}_p = (x_{1p}, x_{2p}, \ldots, x_{np})^T .$$

It is calculated for each neuron its distance D_j (of the vector $\mathbf{w}_j$ from the vector $\mathbf{x}_p$) as an Euclidean distance

$$D_j(\mathbf{w}_j, \mathbf{x}_p) = |\mathbf{w}_j - \mathbf{x}_p|$$

or as a spherical distance

$$D_j(\mathbf{w}_j, \mathbf{x}_p) = 1 - \mathbf{w}_j * \mathbf{x}_p .$$

For the evaluation of D_j is used the formula

$$D_j = \sum_{i=1}^{n} (x_{ip} - w_{ji})^2 \text{, where } j = 1, \ldots, N .$$

The competition of neuron consists in the fact that the neuron is found, that has the weighted coefficients with the smallest distance from values of relevant components of the vector $\mathbf{x}_p$. This neuron is considered to be a winner and it obtains the right of the biggest response $y_{jC} = 1$ and adjusts its weights according to the formula

$$\mathbf{w}_j^{new} = \mathbf{w}_j^{old} + \alpha(\mathbf{x}_p - \mathbf{w}_j^{old})y_j$$

where α is so called learning constant for which applies $0 < \alpha < 1$.
For the winning neuron it applies $y_j = y_{jc} = 1$, then

$$\mathbf{w}_j^{new} = \mathbf{w}_j^{old}(1 - \alpha) + \alpha\mathbf{x}_p$$

and for the loosing neurons it applies $y_j = 0$ and the weight coefficients are not changed with the exception of neurons in a defined surroundings of the winning neuron. Their weights are corrected in a specific way (for example by the same way like for winning neuron).

The surroundings NE_c of the winning neuron is a set of neurons inside the bounded area (for example of a square or (because the circle is an ideal area) the hexagon is used that approximate the circle) around the winning neuron. The radius of surroundings R_C is given by the number of neurons in the surroundings on one side from the winning neuron (for example the radius of square surroundings can be $R_C = 2$ and for hexagon $R_C = 1$). The index C represents the winning neuron.

Probably some other neuron wins when the next learning pattern is set to the input. The process of adjusting of output level of signal, the values of the weight coefficients and surroundings of the new winning neuron is analogue to the first example. The weights of individual neurons are adjusted by consecutive feed of all learning patterns in such a way that certain types of input patterns initiate the neurons in certain places of the network. The neurons are grouped on certain patterns of signals in certain places of network, it is called the map of patterns. When the whole set of learning patterns is used for learning, the network is learned to respond correctly on searched patterns and to distinguish them. For a finer adjustment of the weights it is used the principle of narrowing of the surroundings as well as the downsizing of the value of learning constant. The network is learned in case when further pattern do not change the weight coefficients.

The algorithm of learning of the neural network is as follows:

1) Set up of the number of input variables n according to the structure of input pattern and determination of a number N of efficient neurons according to condition: $L \leq N/(4\log N)$ where L is the number of patterns to be recognized by net.
2) Set up of initial values that are selected as random values in the range $\langle -0.5;0.5 \rangle$.
3) Set up of the shape and radius of initial environs of winning neuron $NE(t_0)$, at first it is greater, major, then it is decreased step by step to be $R_C(t) = 1$.
4) The use of first learning pattern $\mathbf{x}_1$ and foundation of the winning neuron is done in such a way, that the Euclidean distance of its weighted vector $\mathbf{w}_j$ from $\mathbf{x}_1$ is the smallest. The calculation is done according to the formula
$$D_j = \sum_{i=1}^{n} \left(x_i(t) - w_{ji}(t)\right)^2 \rightarrow \min \text{ for } j = 1, \ldots, N$$
The ideal case is the situation when $\min\limits_{j} D_j \rightarrow 0$.
5) The adaptation of weighted coefficients is done according to the formula
$$w_{ji}(t+1) = w_{ji}(t) + \alpha(t)\left(x_i(t) - w_{ji}(t)\right) \text{, for } j \in NE_C(t)$$
and
$$w_{ji}(t+1) = w_{ji}(t) \text{, for } j \notin NE_C(t).$$

The teaching is done so long, until the weighted coefficients do not change if random pattern from the collection of learning patterns is used.

4. Case Study

The case study represents the situation where the coordinates of towns are known and the places of distribution are searched. See tab.1. The places can be searched by calculation of centroids. The centroids have the minimal distances to allocated places. This task can by solved by genetic algorithms and/or neural network.

	Town	X	Y		Town	X	Y
1	London	*0*	*118*	9	Praha	*159*	*76*
2	Paris	*20*	*66*	10	Ljubljana	*159*	*6*
3	Bruxelles	*48*	*98*	11	Zagreb	*177*	*0*
4	Amsterdam	*55*	*121*	12	Wien	*181*	*42*
5	Luxembourg	*64*	*73*	13	Bratislava	*191*	*42*
6	Bern	*75*	*27*	14	Budapest	*213*	*31*
7	Vaduz	*100*	*27*	15	Warszawa	*230*	*117*
8	Berlin	*149*	*118*				

Tab. 1 Coordinates of places

4.1 Case study solved by genetic algorithm

It is used software MATLAB R2008a and its Genetic Algorithm Toolbox to prepare software applications that can be used to solve these types of problems. The input data are represented by coordinates $x_1, x_2, \ldots, x_K$ that characterize the objects. It is possible to define any number of groups. The fitness function represents the sum of squares of distances between the objects and centroids. The coordinates of centroids $c_{j1}, c_{j2}, \ldots, c_{jK}$ $(j=1,2,\ldots,M)$ are changed. The calculation assigns the objects to their centroids. The whole process is repeated until the condition of optimum (minimum) of fitness function is reached. The process of optimization ensures that the defined coordinates $x_{i1}, x_{i2}, \ldots, x_{iK}$ $(i=1,2,\ldots,N)$ of objects and assigned coordinates $c_{j1}, c_{j2}, \ldots, c_{jK}$ of groups have the minimum distances. The fitness function is expressed by following formula

$$f_{\min} = \sum_{i=1}^{N} \min_{j\in(1,2,\ldots,M)} \left(\sqrt{\sum_{l=1}^{K} (x_{il} - c_{jl})^2}\right),$$

where N is the number of objects, M the number of groups and K dimension. Input data are represented by 14 objects with $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ coordinates.

It is convenient to program the task. See the prog.1 called *DPGA.m*. The input data are in an MS Excel format file *DP.xls* and it corresponds to tab.1. The prog.2 *Group.m* is used for calculation of Euclidean distances and prog.3 *Draw.m* is used for drawing the graph.

```
function DPGA
global LOCATION;
num=input('Number of groups:');
num=3*num;
PopSize=input('Population size:');
FitnessFcn = @Group;
numberOfVariables = num;
LOCATION=(xlsread('DP','Coordinates'))
my_plot = @(Options,state,flag) Draw(Options,state,flag,LOCATION,num);
```

```
Options =
gaoptimset('PlotFcns',my_plot,'PopInitRange',[0;300],'PopulationSize',PopSi
ze);
[x,fval] = ga(FitnessFcn,numberOfVariables,Options);
assign=zeros(1,size(LOCATION,1));
for i=1:size(LOCATION,1)
    distances=zeros(num/3,1);
    for j=1:(size(x,2)/3)
        distances(j)=sqrt((LOCATION(i,1)-x(j))^2+(LOCATION(i,2)-
x(size(x,2)/3+j))^2+(LOCATION(i,3)-x(2*size(x,2)/3+j))^2);
    end
    [min_distance,assign(i)]=min(distances);
end
assign
fval
xy=zeros(num/3,3);
for i=1:(num/3)
    xy(i,1)=x(1,i);
    xy(i,2)=x(1,num/3+i);
    xy(i,3)=x(1,2*num/3+i);
end
xy
```

Prog. 1 *DPGA.m*

```
function z=Group(x)
global LOCATION
z=0;
for i=1:size(LOCATION,1)
    for j=1:(size(x,2)/3)
        distances(j)=sqrt((LOCATION(i,1)-x(j))^2+(LOCATION(i,2)-
x(size(x,2)/3+j))^2+(LOCATION(i,3)-x(2*size(x,2)/3+j))^2);
    end
    min_distance=min(distances);
    z=z+min_distance;
end
```

Prog. 2 *Group.m*

```
function state = Draw(Options,state,flag,LOCATION,num)
[unused,i] = min(state.Score);
x=state.Population(i,:);
for i=1:size(LOCATION,1)
    for j=1:(size(x,2)/3)
        distances(j)=sqrt((LOCATION(i,1)-x(j))^2+(LOCATION(i,2)-
x(size(x,2)/3+j))^2+(LOCATION(i,3)-x(2*size(x,2)/3+j))^2);
    end
    [min_distance,assign(i)]=min(distances);
end
for i=1:size(LOCATION,1)
plot3(LOCATION(i,1),LOCATION(i,2),LOCATION(i,3),'sr','MarkerFaceColor',[3*(
assign(i))/num,3*(assign(i))/num,3*(assign(i))/num],'MarkerSize',10);
xlabel('x');ylabel('y');zlabel('z');
grid on;
hold on;
end
plot3(x(1:size(x,2)/3),x((size(x,2)/3+1):2*size(x,2)/3),x(2*size(x,2)/3+1:s
ize(x,2)),'sr','MarkerFaceColor','b','MarkerSize',10);
hold off;
```

Prog. 3 *Draw.m*

The program enables us to set up the number of required groups and the population size. The higher number of individuals the more precise solution but the higher duration of the calculation. Futher, the program sets up the options for optimization and the optimization command *ga* is called. The program involves the calculation of fitness function and it fills the variables with data that inform us about the coordinates of centroids and the assignment of objects to groups and displays them. The two (z-axes is zero) and three dimensional tasks can be solved.

The program is started by command *DPGA* in MATLAB. Then it is necessary to set up the requested number of groups, e.g. *Number of groups* to be *4* and *Population size* to be *1000*.

When the calculation is terminated, the input parameters and results of calculation are displayed on the screen. The results are presented by coordinates of centroids and assignment of places to groups. See Res.1. The fig.1 presents the graph.

```
Number of groups: 4
Population size:1000
assign =
   4   4   4   4   4   2   2   1   1   3   3   3   3   3   1
fval =  415.4
xyz =
 169.9          95.0
   83.1         26.9
 186.4          29.1
   43.5         98.2
```

Res. 1 Coordinates of centroids and location of places

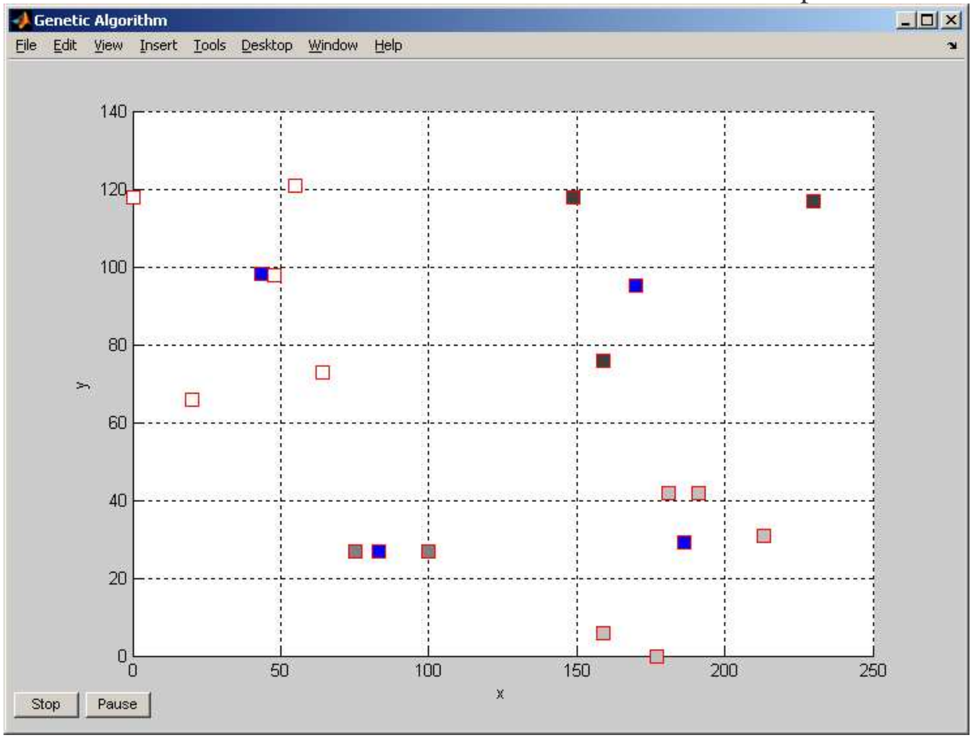

Fig. 1 Places and their centroids

4.2 Case study with neural network

The solution of optimal location of distribution centers by neural network is as follows. It is used software MATLAB 2008a and its Neural Network Toolbox to prepare software applications that can be used to solve these types of problems.

The input data are represented by coordinates $\mathbf{x}_1$, $\mathbf{x}_2,...,\mathbf{x}_k$ that characterize the objects. It is possible to define any number of centroids. The aim is to minimize the sum of squares of distances between the objects (customers) and centroids (distribution centres). The coordinates of centroids $\mathbf{x}_{c1}, \mathbf{x}_{c2},...,\mathbf{x}_{ck}$ are changed. The calculation assigns the objects to their centroids. The whole learning process is repeated until the specified number of learning epochs is reached. The process of learning ensures that the defined coordinates $\mathbf{x}_1$, $\mathbf{x}_2,...,\mathbf{x}_k$ of objects and assigned coordinates $\mathbf{x}_{c1}, \mathbf{x}_{c2},...,\mathbf{x}_{ck}$ of groups have small distances if adequate value of Kohonen parameter is used (in our case 0.1). Input data for a solved task are represented by 14 objects with x_1, x_2 and x_3 coordinates. The input data are in an MS Excel format file *DP.xls* and it corresponds to tab.1.

The program is started by command *DPNN* in MATLAB. See prog.4. The user may choose an arbitrary number of distribution centers and number of iterations, e.g. *Number of groups* to be *4* and *Number of iterations* to be *1000*.

```
clear all;
P=(xlsread('DP','Polohy'))';
num=input('Number og groups:');
iter=input('Number of iterations:');
net=newc([0 230; 0 130; 0 1],num,0.1);
net.trainParam.epochs = 1;
for k=1:iter
net = train(net,P);
xy = net.IW{1};
stem3(xy(:,1),xy(:,2),xy(:,3),'sr','MarkerFaceColor','b','MarkerSize',10)
grid on;
hold on
Q=P';
for i=1:size(Q,1)
    for j=1:(size(xy,1))
        distances(j)=sqrt((Q(i,1)-xy(j,1))^2+(Q(i,2)-xy(j,2))^2+(Q(i,3)-
xy(j,3))^2);
    end
    [min_distances(i),assign(i)]=min(distances);
end
for i=1:size(Q,1)
stem3(Q(i,1),Q(i,2),Q(i,3),'sr','MarkerFaceColor',[assign(i)/num,assign(i)/
num,assign(i)/num],'MarkerSize',10)
xlabel('x');ylabel('y');zlabel('z');
end
figure(gcf)
hold off
end
assign
fval=sum(min_distances)
xy
```

Prog. 4 *DPNN.m*

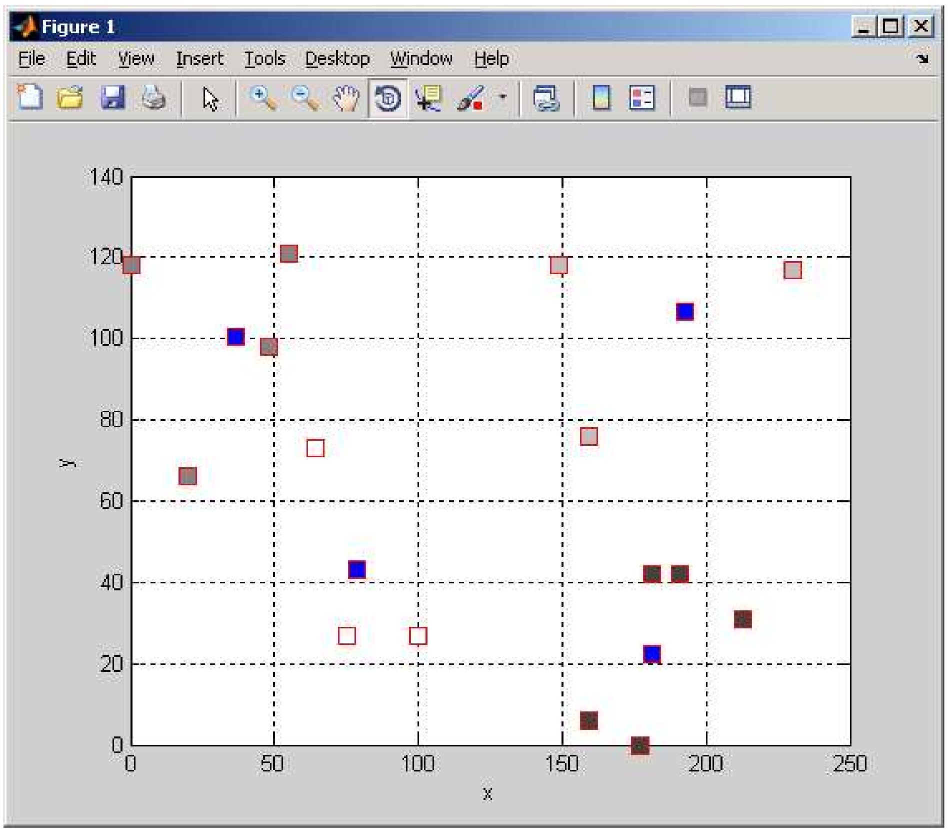

Fig. 2 Places and their centroids

When the calculation is terminated, the input parameters and results of calculation are displayed on the screen. The results are presented by coordinates of centroids and assignment of places to groups. See Res.2. The fig.2 presents the graph.

```
Number og groups:4
Number of iterations:100
assign =2    2    2    2    4    4    4    3    3    1    1    1    1    1    3
fval=448.8
xy =
  181.2          22.5
   36.4         100.4
  192.7         106.5
   78.4          43.0
```

Res. 2 Coordinates of centroids and location of places

5. Conclusion

The genetic algorithms and neural network enable us to solve complicated distribution problems. The correct optimization and application of results in practice enables us to minimize the costs, increase the profit and save our environment. The problem that was solved in practice results in construction of four places of distributions A, B, C and D, that uses places x_1, x_2, …, x_{15}. See fig.3.

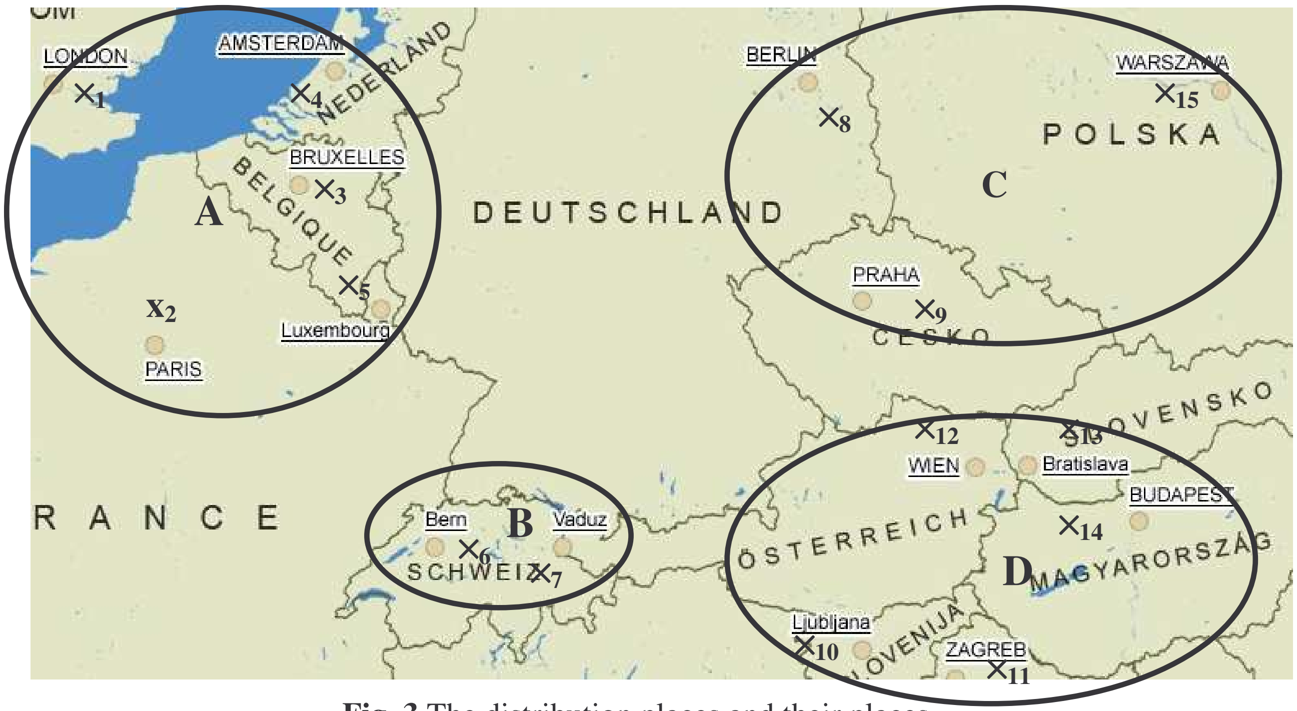

Fig. 3 The distribution places and their places

The distribution problems have a wide range of use in various branches. One of the branches is economy and business. We can mention for example the search of best location of a market, bank or firm. The advantage of the use of genetic algorithms is their applicability in various types of optimization problems with a high speed of calculation and found solution very close to the optimal one. The use of neural networks leads to results that there is a significant dependence of the quality of the results on the chosen value of the Kohonen parameter. To obtain better results for higher number of groups it is better to use higher value of the Kohonen parameter. Finally, the study results in the fact, that for the higher number of centroids the obtained results with use of neural networks are better than results obtained with use of genetic algorithms and vice versa. However, for low number of groups it is better to use genetic algorithms like it is presented in real application in the article.

Genetic algorithms and neural network can be successfully applied in many traditional areas of the operations research where only deterministic models are used. This method can be used for other similar cases and various places.