DOI: 10.46299/ISG.2023.MONO.TECH.3.2.3

2.3 Algorithms for finding the global extremum in the case of the related resources allocation

As studies have shown [58-62], the algorithm described in this article leads to a local extremum. The local minimum is calculated according to the algorithm below:

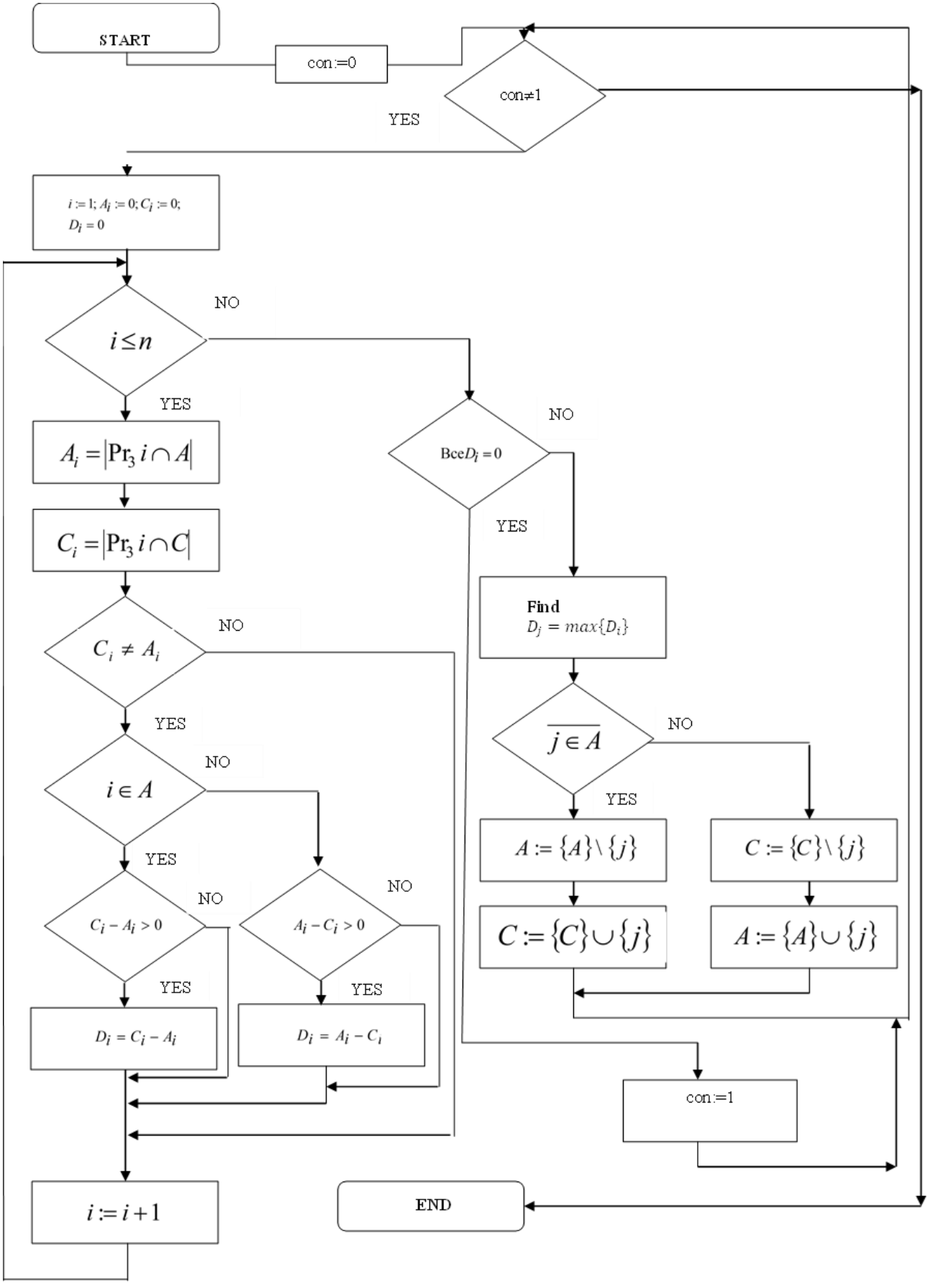
1. An arbitrary initial condition is chosen - the set is cut into two subsets.

2. Third projections of three-dimensional tuples for subsets are considered, i.e. connections in subsets.

3. If the resource (vertex) has more “alien” links than “internal”, such a resource is transferred to the corresponding subset, points 1 and 2 are repeated, i.e. new conditions are considered, etc.

On fig. 1. The local search algorithm is shown, written in the form of a scheme and in the pseudo-language PDL PASCAL.

A local extremum is to be considered found when the further transfer of any resource is not expedient, i.e., a situation will arise when all resources have the number of "internal" neighbors greater than or equal to the number of "alien" neighbors.


START
con:=0
con≠1
YES
$i := 1; A_i := 0; C_i := 0; D_i = 0$
$i \le n$
NO
YES
$A_i = |\mathrm{Pr}_3\, i \cap A|$
$C_i = |\mathrm{Pr}_3\, i \cap C|$
$C_i \neq A_i$
NO
YES
$i \in A$
NO
YES
$C_i - A_i > 0$
NO
YES
$D_i = C_i - A_i$
$A_i - C_i > 0$
NO
YES
$D_i = A_i - C_i$
$i := i + 1$
Все $D_i = 0$
NO
YES
Find
$D_j = max\{D_i\}$
$\overline{j \in A}$
NO
YES
$A := \{A\} \setminus \{j\}$
$C := \{C\} \setminus \{j\}$
$C := \{C\} \cup \{j\}$
$A := \{A\} \cup \{j\}$
con:=1
END

a)

```
Begin
con:=0
 while con ≠ 1 Do
 begin
  begin
   j := 1
   while i ≤ n Do
        begin                    {resetting the original arrays/data sets}
   Aj := 0;
   Cj := 0;
   Dj = 0;
   end
  i = 1
 while i ≤ n Do                  {viewing vertices}
  begin
        Ai = |Pr3i ∩ A|;          {forming "internal" и "alien" connections}
  Ci = |Pr3i ∩ C|;
  IF Ci ≠ A THEN
   begin
   IF i ∈ A THEN
    begin
    IF Ci – Ai > 0 THEN
     begin
     Di := Ci – Ai                {"alien" connections for vertex from A}
     end
    end
   ELSE
    begin
    IF Ai – Ci > 0 THEN
     begin
     Di = Ai – Ci                 {"alien" connections for vertex from C}
     end
    end
   end
   i := i + 1
  end
  IF {для всех i имеет место Di = 0} THEN        {transferring vertex}
   begin
   con := 1                       {no contenders for transfer}
   end
  ELSE                            {contenders for transfer present}
   begin
        {находим j для каждого Dj = max{Di}}     {finding vertex with the peak number of "alien"
                                                  connections}
   IF j ∈ A THEN
    begin
    A := {A} \ {j}                {elimination j from A}
    C := {C} ∪ {j}                {inserting j into array C}
    end
   ELSE
    begin
    C := {C} \ {j};
    A := {A} ∪ {j}
    end
   end
  end
End
```

б)

Figure 1. Algorithm for finding a local minimum: a). structural chart, б). in the pseudo-language PDL PASCAL

The problem of finding a global extremum arises. To construct a systematic approach to solving this problem, we use the definitions introduced in [63].

An individual optimization problem is a pair (F,c), where F – arbitrary set, area of admissible points, and c – cost function that maps c: $F \to R^1$. The point $f \in F$ *is to be found,* for which $c(f) \le c(y)$ for all $y \in F$. This point f named in [63] *a globally optimal solution for a given individual problem, or, where there can be no misunderstanding, simply the optimal solution.*

An optimization problem is a set of ***I*** *individual optimization problems. This two problems are different. An individual problem is given "input data" and has enough information to obtain a solution, while a problem is a set of individual problems, usually generated in the same way.* If a valid point $f \in F$ is given in a certain specific problem, then in many cases it is useful to define a set N(f) of points that are in some sense "close" to a given point f. Let an optimization problem with a set of individual problems (F,c) be given. Then the vicinity system (or vicinity function) is the map N: $F \to 2^F$, specific for each individual problem.

If $F=R^n$, then the set of points lying within a fixed Euclidean distance from the given point forms a natural vicinity. In many combinatorial problems, the choice of N can significantly depend on the structure of F. Returning to the question of finding a global solution, it should be noted that it is often possible to find a solution f that is best insofar that there is nothing better in its vicinity N(f).

Let an individual optimization problem (F,c) and a system of vicinities N, then a feasible solution $f \in F$ is called locally optimal with respect to N (or just locally optimal when N is obvious from the context), if $c(f) \le c(g)$ for every $g \in N(f)$.

In the general case, the search algorithm for a global solution (global extremum) is reduced to repeated application of the local extremum search algorithm (LESA) under different initial conditions and choosing the best solution from those obtained.

Thus, the algorithm for searching a global extremum should be based on 2 basic algorithms:

an algorithm for determining a new initial condition (variant) (ADIC) that takes the vicinity of a local extremum (LE) out of the area;

LESA in a new vicinity (vicinity of a new LE).

LESA as applied to the graph model of the allocation of related resources problem for the case, when $K = 2$, p_j limitations are absent, and all $w_{ij} = 1$.

The generalized local search algorithm is shown in Figure 2.

```
begin
    x^0:= certain initial option of the resolution;
    t:= 0; h:=0;
    while { the ambit of the current option has not been considered Sε(x^h)} do
    begin
        y:= another option from Sε(x^h);
        Δ:= R(x^h) – R(y);
        if (Δ>0) then
        begin
            h:=h+1;  {increase  pace}
            x^(h+1):=y; {move to a new point}
        end;
        t:=t+1;
    end;
    withdraw x^h;
end;
```

Figure 2. The generalized local search algorithm (LS).

Metrics vicinity:

$$S\varepsilon(x)=\{y\in X: d(x,y)\le\varepsilon\}$$

Metrics:

$$d(x,y)=\sum_{k=1}^{m}|x_k - y_k|,\ \forall x,y \in X.$$

On fig. 3 - 5, 7 various LESA patterns are shown.

The general algorithm of local search with repetition is shown on fig. 3.

```
begin
    x:= certain initial option of the resolution;
    x̄ :=local search of x;
    t:=0; h:= 0;
    while {shutdown predicate not satisfied} do
```

```
begin
    x̃ :=disturbance x̄ ;
    x̿ :=local search x̃ ;
    if {recipiency predicate is satisfied} then
    begin
        x̄ := x̿ ;
        h:=h+1;
    end;
    t:=t+1;
 end;
 withdraw x̄ ;
end;
```

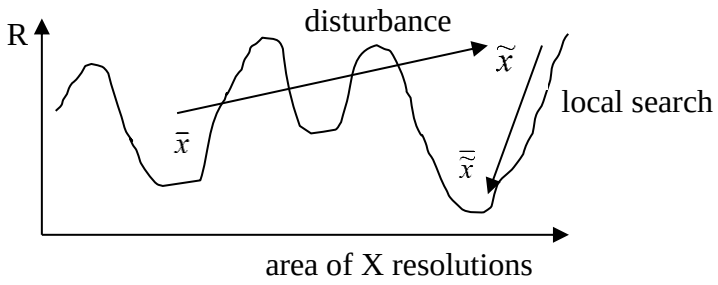


Figure 3. The general algorithm of local search with repetition (LSR).

The general algorithm for recoil simulation based on the Boltzmann principle is shown on Fig. 4.

begin

x^0:= certain initial option of the resolution;

T_0:= initial temperature {conditional indicator, changing according to the Boltzmann-Gibbs formula - $p=e^{-\Delta/T}$};

t:=0; h:=0;

x_{rec}:=x^0; R_{rec}= R(x_{rec});

while {stop predicate is not veritable} do

begin

while {balance condition not satisfied} do

begin

y:= certain variant of $S_\varepsilon(x^h)$;

Δ:=R(y) - R(x^h);

p:=min{1, $e^{-\Delta/T}$};

ξ:=random[57,58] **{random number generation};**

if (p>ξ) then

begin

x^{h+1}:=y;

replace if necessary x_{rec}, R_{rec};

h:=h+1;

end;

end;

withdraw x_{rec};

end;

Figure 4. The general algorithm for recoil simulation based on the Boltzmann-Gibbs principle (RS)

The general algorithm for accelerated probabilistic modeling is shown on fig. 5.

begin

x^0:= certain initial option of the resolution;

μ_0:=0;

t:=0; h:=0;

x_{rec}:=x^0; R_{rec}:= R(x_{rec});

while {stop predicate verity condition not satisfied} do

begin

while {balance condition not satisfied} do

begin

y:= certain variant of $S_\varepsilon(x^h)$;

Δ:= R(y) – R(x^h);

p:=min{1, 1 - Δ/(γR(x^h)};

ξ:= μ_t + random[57, 58] **(1 - μ_t);**

if (p>ξ) then

begin

x^{h+1}:=y;

replace if necessary x_{rec}, R_{rec};

h:=h+1;

end;

end;

formulate the next value μ_{t+1};

t:=t+1;

end;

withdraw x_{rec};

end;

Figure 5. The general algorithm for accelerated probabilistic modeling (APM)

To solve the problem of finding a global extremum, the idea of heuristic self-organization or, developed on its basis, the strategy of "genetic algorithm" has been

actively used recently. The general pattern of the genetic algorithm is shown on Figure 7.

Fig. 8 shows the results of the study of the convergence of the algorithms above.

With regard to the studied problem in terms of a genetic algorithm, recalling the formula

$$y_i = \begin{cases} 0 & if\ x_i \in A \\ 1 & if\ x_i \in C. \end{cases} \qquad i = \overline{1, N},$$

a general view of the "chromosome" is obtained.

y_1	y_2	y_3	...	y_i	...	y_N

In fact, this is the characteristic vector of the partition.

The next chromosome describes the partition shown on fig. 6

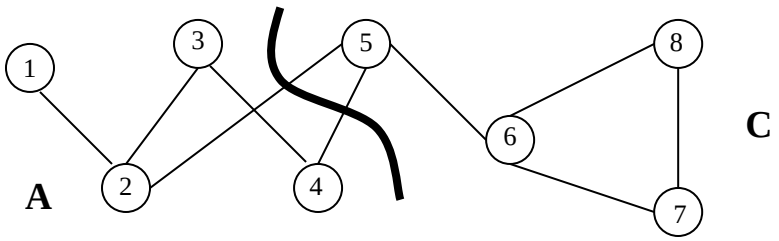


Figure 6. Graph diagram

1	**1**	**1**	**1**	**0**	**0**	**0**	**0**

The genes that make up the original population are separate vector-orts, in which the i-th position contains the value y_i, that fixes the affiliation of the i-th vertex to the set A ($y_i = 1$)

			...	**1**	...	

or to the set C ($y_i = 0$).

			...	**0**	...	

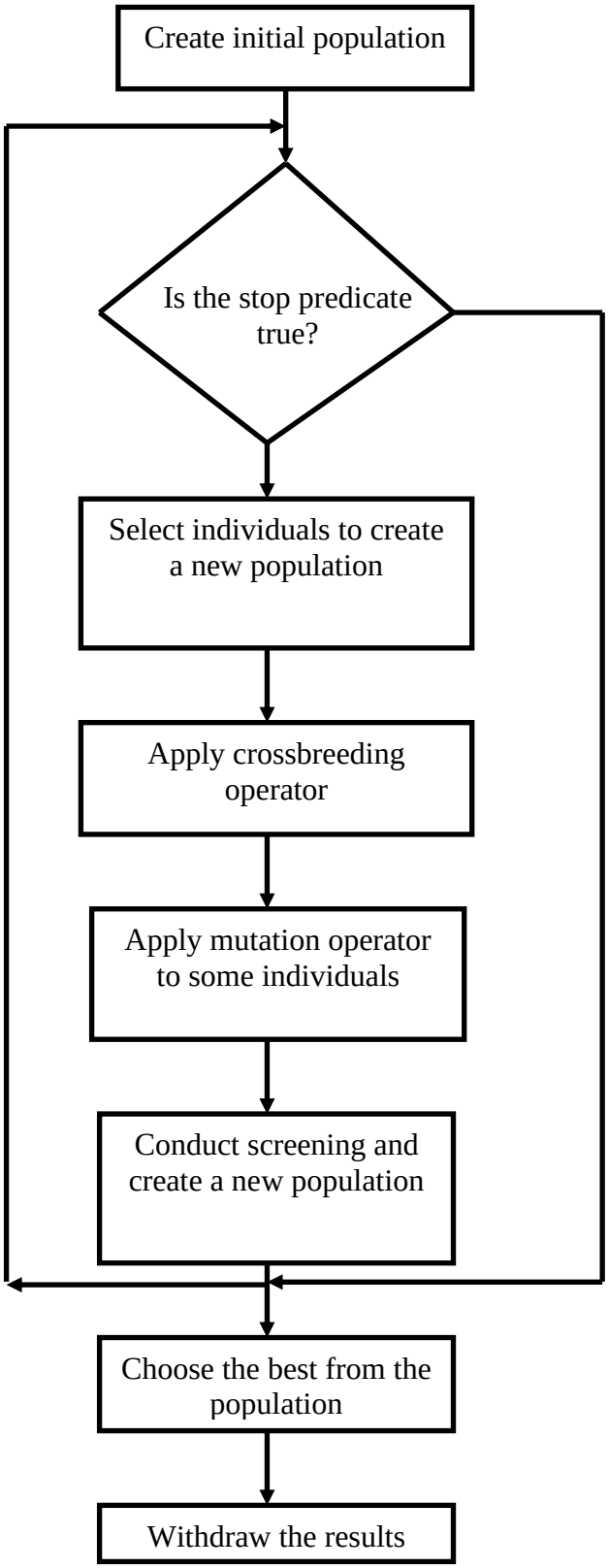
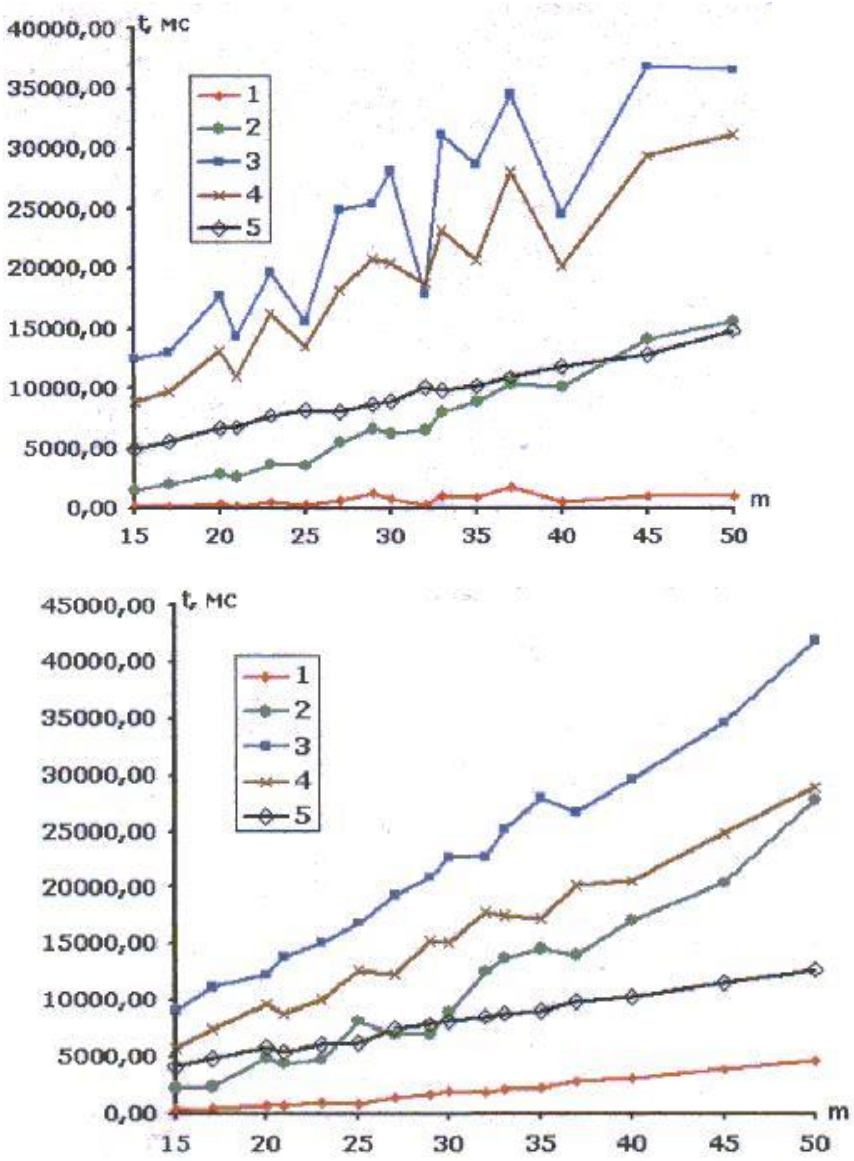


Figure 7. General pattern of the genetic algorithm (GA)

The rest of the vector components are not defined.



1 - LS, 2 - LSR , 3 - RS , 4 - APM, 5 - GA.

Figure 8. Results of the study of global extremum search algorithms.

The elimination of unpromising chromosomes occurs according to the criterion of minimality of “broken” bonds (between resources that fell into different sets):

$$\sum_{i=1}^{n}\sum_{j<i} w_{ij}\left|y_i - y_j\right| \longrightarrow \min$$

or

$$\sum_{i=1}^{n}\sum_{j<i} w_{ij} y_i (1-y_j) + \sum_{i=1}^{n}\sum_{j<i} w_{ij} (1-y_i) y_j \longrightarrow \min.$$

Combinations of genes describing the clique of the original graph should be considered promising, because it is assumed that the rapture of such a clique leads to an increase in the number of broken bonds.

The idea of genetic algorithms does not change the heuristics of the applied rules, therefore, is not to be used for this research in the future.