ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol.34, No.1

PERFORMANCE OF HYBRID GENETIC ALGORITHM FOR THE GREY PATTERN PROBLEM[♣]

Alfonsas Misevičius, Dalius Rubliauskas

Kaunas University of Technology, Department of Practical Informatics
Studentų St. 50–416a, LT–51368 Kaunas, Lithuania

Abstract. Recently, genetic algorithms (GAs) are quite popular by solving combinatorial optimization problems. In this paper, we discuss a hybrid genetic algorithm that uses a new kind of solution recombination operators – a so-called multiple parent crossover. We examined this innovative crossover operator on the grey pattern problem, which is as special case of the well-known problem, the quadratic assignment problem. The results obtained during the experimentation with the set of 62 instances of the grey pattern problem demonstrate promising efficiency of the multiple parent crossover. All the instances tested were solved to pseudo-optimality within surprisingly small computation times.

Keywords: combinatorial optimization, heuristic algorithms, genetic algorithms, multiple parent crossover, grey pattern problem.

Indroduction

The grey pattern problem – firstly introduced by Taillard [22] – is based on a rectangle (grid) of dimensions $n_1 \times n_2$ containing $n = n_1 \times n_2$ points (square cases) with m black points (cases) and $n - m$ white points. By juxtaposing many of these rectangles, one gets a grey pattern (frame) of density m/n. The objective is to get the finest grey pattern, that is, the black points have to be spread on the rectangle as regularly as possible. The grey pattern problem is a special case of a more general problem, the quadratic assignment problem (QAP) [11] which is known to be NP-hard. The QAP is formulated in the following way. Let two matrices $\boldsymbol{A} = (a_{ij})_{n\times n}$ and $\boldsymbol{B} = (b_{kl})_{n\times n}$ and the set Π of all possible permutations of the integers from 1 to n be given. The goal is to find a permutation $\pi = (\pi(1), \pi(2), ..., \pi(n)) \in \Pi$ that minimizes

$$z(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}. \qquad (1)$$

In the grey pattern problem (as formulated in [22]), the matrix $(a_{ij})_{n\times n}$ is defined as $a_{ij} = 1$ for $i, j = 1, 2, ..., m$ and $a_{ij} = 0$ otherwise. The matrix $(b_{kl})_{n\times n}$ is defined by the given values – the distances between every two of n points. More precisely, $b_{kl} = b_{n_2(r-1)+s\ n_2(t-1)+u} = f_{rstu}$, where

$$f_{rstu} = \max_{v,w\in\{-1,0,1\}} \frac{1}{(r-t+n_1 v)^2 + (s-u+n_2 w)^2},$$

$r, t = 1, ..., n_1$, $s, u = 1, ..., n_2$. f_{rstu} may be thought of as an electrical repulsion force between two electrons (to be put on the grid points) i and j ($i, j = 1, ..., n$) located in the positions $k = \pi(i)$ and $l = \pi(j)$ with the coordinates (r, s) and (t, u). The ith ($i \le m$) element of the permutation (solution) π, $\pi(i) = n_2(r - 1) + s$, gives the location in the rectangle (grid) where a black point (case) has to be placed in. The coordinates of the location $\pi(i)$ are derived according to the formulas: $r = \lfloor(\pi(i) - 1)/n_2\rfloor + 1$, $s = ((\pi(i) - 1) \bmod n_2) + 1$, $i \le m$. See [22] for more details. One can obtain plenty of different instances of the grey pattern problem by varying the choices of n_1, n_2 and m (see Section 3).

Many heuristic approaches may be applied for solving both the QAP and, at that time, its particular case – the grey pattern problem. For serveys of the heuristics for the QAP, see [2,3,22]. Genetic algorithms and their hybrids have recently achieved great success in solving the QAP [1,5,12,14,15]. In this paper, we propose, a hybrid genetic algorithm (HGA) which incorporates an innovative operator of the recombination of solutions. The template of this algorithm and the details of the new recombination (crossover) operator are discussed in Section 2. The computational results of HGA for the various grey pattern problem instances are presented in Section 3. Section 4 completes the paper with concluding remarks.

[♣] This work is supported by Lithuanian State Science and Studies Foundation.

2. Hybrid genetic algorithm using multiple parent crossover for the grey pattern problem

The original concepts of genetic algorithms (GAs) were developed by Holland [10] in 1975. A genetic algorithm operates with a group P (called a population) of solutions s_1, s_2, ..., $s_{PS=|P|}$ (called individuals) from S − the set of solutions of a combinatorial optimization problem. Each individual (s_i) is associated with some fitness, i.e. the objective function value ($f(s_i)$). In minimization problems, the less the objective function value, the more fitting the individual, and the larger is the probability that the individual will survive in evolution process. During many generations, best fitting individuals tend to dominate, while less fitting ones tend to die off.

The main components of GA are as follows [4, 8, 21]: a) a mechanism of selecting individuals from the population; b) an operator for creation new solutions by combining pairs of previous solutions (i.e. "parents") (this operator is known as a crossover); c) a mutation procedure that generates new solutions by random perturbations of the existing solutions; d) a population replacement (culling) scheme. Our focus is on the crossover operator, which is responsible for the efficiency of the genetic search in a high degree. It should be noted that the state-of-the-art genetic algorithms are rather hybrid algorithms which incorporate additional heuristic components [19]. In such algorithms there is used a post-crossover procedure that play the role of a local improvement algorithm applied to the solution produced by the crossover. However, this fact does not imply that the performance of recombination operators is not important anymore. In this paper, we propose an improvement of HGA which is exactly due to the new enhanced crossover operator.

Dozens of crossover operators for the permutation-based problems are known from the literature, for example, cohesive crossover [5], cycle crossover [20], distance preserving crossover [13], partially mapped crossover [9], uniform (like) crossover [24], and many others. As a rule, these crossovers share one principle characteristic: the offspring is created by using two parents. There are two aspects of these crossovers. Firstly, they are distinguished for the conceptual simplicity and relatively high efficiency in solving such problems, like the quadratic assignment problem or the traveling salesman problem. On the other hand, some shortcomings of the typical crossover operators might be discovered. For example, some useful information may be left out of account by using two parents only. The other negative aspect is related to the fact that there exists a quite large degree of randomness. This is especially true when parents are selected in a pure random manner. In this case, it is obvious that the parents will, most probably, produce a "child" of rather poor quality. This kind of behaviour may be viewed as one of the most pessimistic factors related to the traditional crossover operators. In order to try to overcome these difficulties, innovative crossover procedures should be proposed. In this paper, we introduce such a non-ordinary crossover − we call it "μ-crossover" (or shortly MX). This name can be thought of as a derivative from the term "multiple parent crossover", i.e. "crossover based on μ parents". So, the heart of the new crossover is generation of the offspring by means of several parents. The details of this crossover are discussed below.

The main criterion for high quality MX operator is the ability of inheritance of the features contained in all the parents that take part in generation of the offspring. Which way we can implement this criterion? Our idea is to use so-called desirability measures for the elements of a solution[*]. Let μ be the number of the solutions-parents (i.e. chromosomes in the context of GA) to produce a solution-offspring. The solution is organized as a certain permutation $\pi = (\pi(1), \pi(2), ..., \pi(n))$, where $\pi(i)$ denotes the position (also called a locus) that the element i (also called a gene) is assigned to. Then, the desirability information can be maintained in a matrix $\boldsymbol{D}$ of size $n \times n$, where the entry d_{ij} is simply equal to the number of times that the element, i.e. gene i is assigned to the position, i.e. locus $j = \pi(i)$ in the parents (i.e. the set that consists of μ chromosomes). The following are the simple properties of the entries of $\boldsymbol{D}$: 1) $0 \le d_{ij} \le \mu$, $i, j = 1, 2, ..., n$; 2) $\sum_{i=1}^{n} d_{ij} = \mu$, $j = 1, 2, ..., n$; 3) $\sum_{j=1}^{n} d_{ij} = \mu$, $i = 1, 2, ..., n$. Naturally, the larger the value of d_{ij}, the more is desirable that $\pi(i)$ is set to j ($\pi(i) = j$) in the offspring. Let $\Pr(\pi_{\text{offspr}}(i) = j)$ denote the probability that the gene i will be assigned to the locus j in the offspring π_{offspr}. We assume that this probability is equal to the aspect ratio $\frac{\text{number of times that } \pi(i) = j}{\text{number of parents}}$, that is, $\Pr(\pi_{\text{offspr}}(i) = j) = \frac{d_{ij}}{\mu}$. Then, it is obvious from the properties (2), (3) that: 1) $\sum_{i=1}^{n} \Pr(\pi_{\text{offspr}}(i) = j) = 1$, $j = 1, 2, ..., n$ (this means that, in the offspring's chromosome, every locus will necessary be associated with one of the genes); 2) similarly, $\sum_{j=1}^{n} \Pr(\pi_{\text{offspr}}(i) = j) = 1$, $i = 1, 2, ..., n$ (this means that every gene will be associated with one of the loci). Taking the above facts into account, a natural way to create a gene i (i.e. to obtain a locus for the current gene i) is to choose such a number j (among those not yet chosen) that $\Pr(\pi_{\text{offspr}}(i) = j)$ is maximized. (Of

[*] Note that this idea has some similarities with the adaptive memory principle (see [7] for more details).

course, if the gene is assigned to the same locus in all the parents, then this gene remains at the same locus for the offspring.) This process is to be continued until all the genes are assigned to their loci. The detailed template of the resulting multiple parent crossover procedure (in the *Pascal*-language like notation) is presented Figure 1. The memory size and time complexity of this crossover is $O(n^2)$. An illustrative example of MX is shown in Figure 2. It should be noted that MX offers some degree of randomization. Randomness is achieved by the existence of many variants for choosing different sequences of the genes.

```
function MX($parents$, $\mu$);  // $\mu$-crossover //
   // input: $parents$ – the structure, i.e. the matrix containing the parents, $\mu$ – the number of parents //
   // output: $\pi$ – the resulting offspring (permutation) produced by $\mu$ parents //
   $D := 0$;
   for $i := 1$ to $\mu$ do for $j := 1$ to $n$ do $D(j, parents(i,j)) := D(j, parents(i,j)) + 1$;
   $I := \{1, 2, ..., n\}$;  $J := \varnothing$;
   repeat  // continue until the offspring is created //
      choose $i \in I$;
      $\pi(i) := \arg\max_{j=1,2,...,n;\ j \notin J} D(i, j)$ ;
      // ties (i.e. situations when more than one $j$ satisfying the given equation exist) are broken in a random way //
      $I := I \setminus \{ i \}$;  $J := J \cup \{ \pi(i) \}$
   until $I = \varnothing$;
   return $\pi$
end.
```

Figure 1. Pseudo-code of the template of the μ-crossover

Current population (population size: 5)

4	3	6	7	1	2	9	8	5
4	3	6	7	1	9	5	8	2
4	6	3	1	7	5	9	2	8
4	7	3	1	8	5	9	6	2
5	6	3	1	2	4	9	7	8

Desirability matrix

0	0	0	4	1	0	0	0	0
0	0	2	0	0	2	1	0	0
0	0	3	0	0	2	0	0	0
3	0	0	0	0	0	2	0	0
2	1	0	0	0	0	1	1	0
0	1	0	1	2	0	0	0	1
0	0	0	0	1	0	0	0	4
0	1	0	0	0	1	1	2	0
0	2	0	0	1	0	0	2	0

Suppose that the genes are picked up in the following order: 7, 3, 1, 8, 2, 6, 5, 4, 9.

Then, the offspring's chromosome is created as follows:

$$\pi(7) = \arg\max_{j}\{\Pr(\pi(7) = j)\} = \arg\max_{j}\{d_{7j}\} = 9 ;$$

$$\pi(3) = \arg\max_{j \neq 9}\{\Pr(\pi(3) = j)\} = 3 ;$$

$$\pi(1) = \arg\max_{j \neq 3,9}\{\Pr(\pi(1) = j)\} = 4 ;$$

Offspring

4	**6**	**3**	**7**	**1**	**5**	**9**	**8**	**2**

Figure 2. Example of producing of the offspring in μ-crossover ($\mu = 5$)

The multiple parent crossover distinguishes oneself for the one more important feature. As long as the number of parents in the crossover, μ, is equal to the current population size, *PS*, there no need in any selection procedure: the selection is "hidden" in the crossover itself; in the other words, MX performs the functions of both selection and recombination. This variant (i.e. $\mu = PS$) was used in our implementation.

In order to increase the performance of GA even more, the crossover can be applied more than once at the same generation. In our implementation, the number of MXs per one generation is controlled by the parameter N_{offspr} (# of offspring per generation).

The remaining components of the hybrid genetic algorithm for the grey pattern problem are identical to those of HGA for the quadratic assignment problem, except the specific cases discussed below. The framework of this algorithm (entitled as HGA-MX-TS) is presented in Figure 3. The details can be found in [17]. Remind that the outstanding performance of HGA for the QAP was achieved by exploiting the idea of genetic-tabu search, i.e. combining the genetic operators with the enhanced tabu search (TS) procedure –

as a local improvement (post-crossover) algorithm. The details of the TS procedure are described in [18]. One important modification should be mentioned. It is related to the performance of the tabu search, more precisely, the exploration of neighbourhoods (i.e. the sets of neighbouring solutions of the current solutions), as well as the calculation of the differences in the objective function values. A lot of the computations can be shorten and simplified (consequently, the large amount of computer's (CPU) time may be saved) due to the very special character of the matrix ***A*** in the grey pattern problem, as shown in [22]. For this problem, the exploration of the neighbourhood in the TS procedure is restricted to the interchange of one of the first *m* elements (black points) with one of the last $n - m$ elements (white points). Therefore, the neighbourhood size decreases to $O(m(n-m))$, instead of $O(n^2)$ for the ordinary QAP. In addition, the calculation of the differences in the objective function values becomes more faster because the matrix ***A*** is consisting of entries 0 and 1 only. So, instead of the standard formula of calculation difference in the objective function values when exchanging the *i*th and *j*th elements in the current permutation

$$\begin{aligned}\Delta z(\pi,i,j) = {} & (a_{ii}-a_{jj})(b_{\pi(j)\pi(j)}-b_{\pi(i)\pi(i)}) + \\ & (a_{ij}-a_{ji})(b_{\pi(j)\pi(i)}-b_{\pi(i)\pi(j)}) + \\ & \sum_{k=1,k\neq i,j}^{n}(a_{ik}-a_{jk})(b_{\pi(j)\pi(k)}-b_{\pi(i)\pi(k)}) + \\ & \sum_{k=1,k\neq i,j}^{n}(a_{ki}-a_{kj})(b_{\pi(k)\pi(j)}-b_{\pi(k)\pi(i)}), \\ & i=1,2,...,n-1,\ j=i+1,...,n,\end{aligned} \tag{2}$$

the simplified formula

$$\Delta z(\pi,i,j) = 2\sum_{k=1,k\neq i}^{m}(b_{\pi(j)\pi(k)}-b_{\pi(i)\pi(k)}),$$
$$i=1,2,...m,\ j=m+1,....n \tag{3}$$

is used. As a result, the TS algorithm complexity is reduced from $O(n^3)$ to $O(m^2(n-m))$. As the TS procedure is invoked many times during the execution of HGA, the overall effect is even more evident, especially, in the cases when $m \ll n$. All these favourable circumstances allowed to treat very large problems ($n = 256$) with reasonable CPU times (see Section 3).

```
function HGA-MX-TS($A$, $B$, $n$); // hybrid genetic algorithm using MX-crossover and tabu search //
    // input: $A$, $B$ – the matrices, $n$ – the problem size; output: $\pi^*$ – the best solution (permutation) found //
    // parameters: $PS$ – the population size, $N_{gen}$ – # of generations, $N_{offspr}$ – # of offspring per generation, //
    //              $\mu$ – the number of parents //
    read $A$, $B$, $n$, $PS$, $N_{gen}$, $N_{offspr}$, $\mu$;
    create the initial population $P \subset \Pi$, where $|P| = PS$;
    $\pi^* := \operatorname{argmin}_{\pi \in P} z(\pi)$ ; // $\pi^*$ denotes the best so far solution //
    for $generation := 1$ to $N_{gen}$ do begin // generations cycle //
      for $child := 1$ to $N_{offspr}$ do begin // offspring creation cycle //
        select $\mu$ solutions, i.e. parents from $P$: these solutions are organized
        as $\mu \times n$ matrix entitled $parents$, where $parents(i)$ denotes the $i$th parent,
        and $parents(i,j)$ is the $j$th element in the $i$th parent;
        $\dot{\pi} :=$ MX($parents$, $\mu$); // the offspring is created by applying the multiple parents crossover to $\mu$ parents //
        improve $\dot{\pi}$ by using tabu search, get the resulting solution $\pi^\bullet$;
        add the improved permutation $\pi^\bullet$ to the population $P$;
        if $z(\pi^\bullet) < z(\pi^*)$ then $\pi^* := \pi^\bullet$ // save the best so far solution //
      end; // for $child$ ... //
      cull the population $P$ by removing $N_{offspr}$ worst individuals;
      if the level of diversity of $P$ is below the predefined threshold then
        make a "restart"
    end; // for $generation$ ... //
    return $\pi^*$
end.
```

Figure 3. Pseudo-code of the template of the hybrid genetic algorithm

3. Computational experiments

Before extensive testing of HGA-MX-TS on the large grey pattern problems, we have conducted a small experiment to demonstrate the behaviour of the new proposed crossover operator. (A data instance with $n = 9$ compiled by A. Misevičius was used in this experimentation.) To show the possible benefits of the multiple parent crossover, we compared the results produced by our new crossover and the traditional two-parent crossover — namely the uniform (like) crossover due to Tate and Smith [24] (so far, this crossover and its modifications have been proven to be quite efficient). The results obtained from this experiment are presented in Figure 4. They confirm the "aggressiveness" of the multiple parent crossover. It can be seen that MX enables to explore the solution space and to direct the search in promising regions quite efficiently. Multiple parents seem to be able to discover the "building blocks" — these blocks are of the highest importance in genetic search — surprisingly effectively. This can be seen clearly when comparing the results of MX and the two-parent crossover.

Further, the more thorough computational experiments have been carried out on a set of 62 instances of the grey pattern problem. For the instance family tested, the size of the instances is equal to 256, and the frames are of dimensions 16×16. The parameter m, i.e. the density of grey varies from 3 to 64. The instances are denoted by the name grey_16_16_*m*, where $2 \le m \le 64$. Remind that, for these instances, the data matrix $\boldsymbol{B}$ remains unchanged, while the data matrix $\boldsymbol{A}$ is of the form $\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, where $\mathbf{1}$ is a submatrix of size $m \times m$ composed of 1s only [23]. All these instances were examined by our hybrid genetic algorithm with the multiple parent crossover. The goal was to find out how difficult are the grey pattern problems for HGA (which has been proven to be extremely efficient for the QAP) and, especially, for MX. We focused on the average computation time needed to find the pseudo-optimal (best know) solutions for these problems. Note that, for many instances, we performed several independent runs each consisting of 10 restarts of HGA-MX-TS. Various combinations of the values of the control parameters, which depend on the particular instance, are used in the different runs. The best CPU times obtained during these runs are given in Table 1. The ranges of the main parameter values for HGA-MX-TS are as follows: *PS* varies between 4 and 30; N_{gen} – between 2 and 100; $N_{\text{offspr}} = 1$. The number of parents in MX is equal to the population size *PS* in all the experiments.

It can be viewed from Table 1 that the efficiency of HGA-MX-TS for the grey pattern problem is very promising. The results indicate that all the instances examined are most probably solved pseudo-optimally at really short CPU times. (There were only a negligible number of instances with relatively large CPU times. So far, we have no well-founded explanations of these anomalies, except that the algorithm sometimes tends to converge (in fact, misconverge) to high-quality locally optimal solutions which may be quite "far" from a global optimum.) The performance of HGA-MX-TS for the particular instances is impressive indeed. For example, for the largest instance tested grey_16_16_64 ($m = 64$), less than 5 seconds of 3GHz Pentium computer are enough to find a pseudo-optimal solution. We guess that the search time may be decreased even more by a careful tuning of the control parameters of HGA-MX-TS.

The quality of the solutions obtained is also confirmed by the graphical illustrations. In Figure 5, we give twelve frames that correspond to the pseudo-optimal solutions of the instances grey_16_16_53.. grey_16_16_64. So, the reader can grasp the quality of the obtained solutions from the visual point of view, too.

(a)

Initial (improved) population

4	7	2	6	3	9	1	8	5	121476
4	6	**2**	5	7	3	9	**1**	8	124142
4	3	9	8	1	5	7	2	6	125376
2	5	7	8	6	3	9	1	4	125578
8	7	9	2	4	5	6	3	1	132120

1st offspring

4	7	**2**	8	3	**5**	9	**1**	6

Population after crossover and improvement: 2nd generation

4	7	**2**	8	6	3	9	**1**	5	121034
4	7	2	6	3	9	1	8	5	121476
4	6	**2**	5	7	3	9	**1**	8	124142
4	3	9	8	1	5	7	2	6	125376
2	5	7	8	6	3	9	1	4	125578

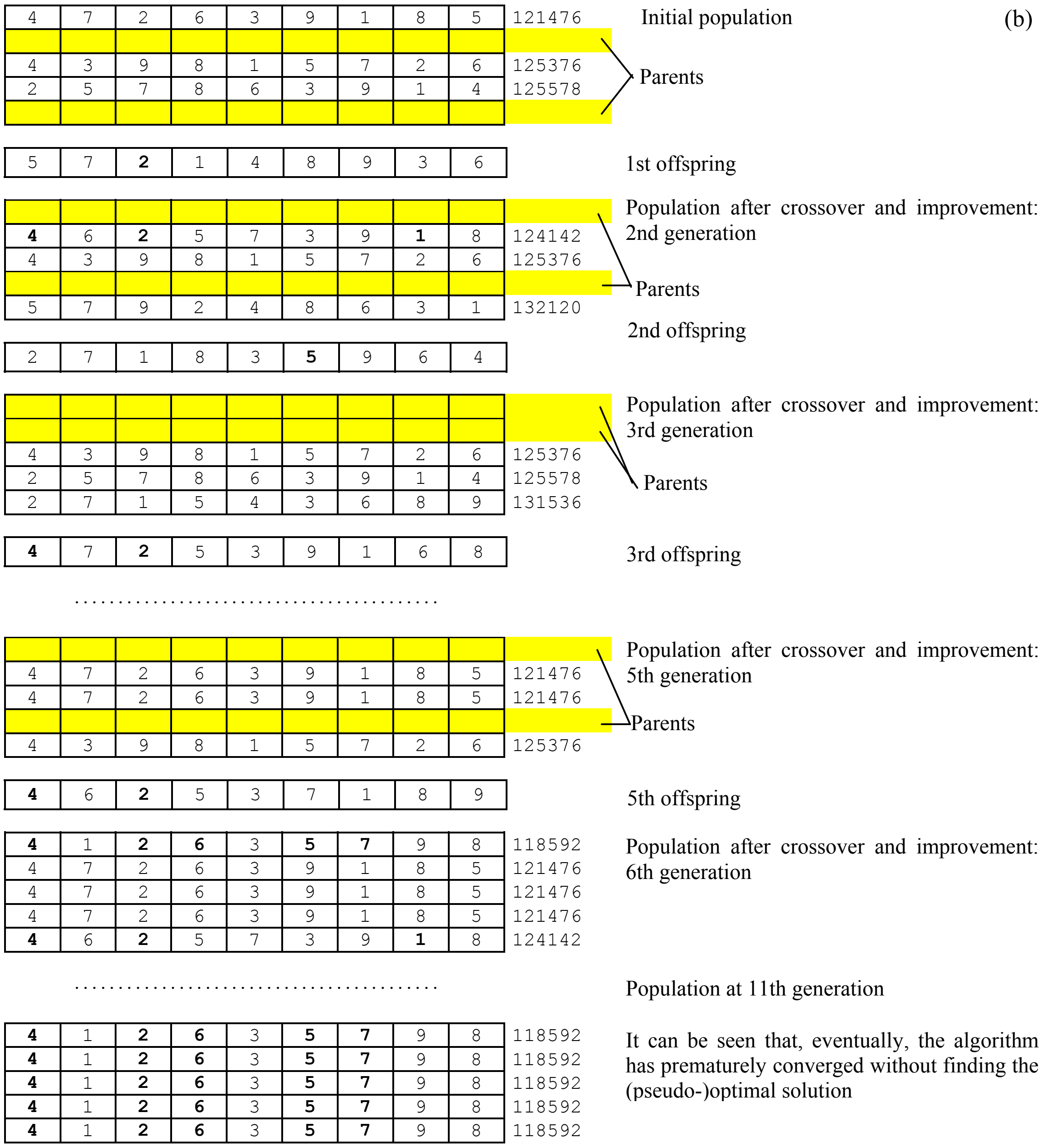
2nd offspring

4	7	**2**	8	6	3	9	**1**	5

Population after crossover and improvement: 3rd generation

4	**9**	**2**	**6**	**8**	**5**	**7**	**1**	**3**	**118180**
4	7	2	8	6	3	9	1	5	121034
4	7	2	6	3	9	1	8	5	121476
4	6	2	5	7	3	9	1	8	124142
4	3	9	8	1	5	7	2	6	125376

(Pseudo-)optimal solution


(b)
Initial population
Parents
1st offspring
Population after crossover and improvement: 2nd generation
Parents
2nd offspring
Population after crossover and improvement: 3rd generation
Parents
3rd offspring
Population after crossover and improvement: 5th generation
Parents
5th offspring
Population after crossover and improvement: 6th generation
Population at 11th generation

It can be seen that, eventually, the algorithm has prematurely converged without finding the (pseudo-)optimal solution

Figure 4. Comparison of the genetic processes by using the multiple parent crossover (a) and 2-parent crossover (b). Note. The elements (items) that correspond to "building blocks" (i.e. the elements that are likely to be contained in the optimal solution) are printed in bold face

Table 1. Results of the experiments for grey pattern problems

Instance name	Best known value	Time[‡]	Instance name	Best known value	Time[‡]
grey_16_16_3	7810[❖]	*0.0*	**grey_16_16_34**	4560162[a]	*5.1*
grey_16_16_4	15620[❖]	*0.0*	**grey_16_16_35**	4890132[a]	*6.3*
grey_16_16_5	38072[❖]	*0.0*	**grey_16_16_36**	5222296[a]	*3.4*
grey_16_16_6	63508[❖]	*0.0*	**grey_16_16_37**	5565236[a]	*3.9*
grey_16_16_7	97178[❖]	*0.0*	**grey_16_16_38**	5909202[a]	*1.6*
grey_16_16_8	131240[❖]	*0.0*	**grey_16_16_39**	6262248[a]	*1.9*
grey_16_16_9	183744[a]	*0.1*	**grey_16_16_40**	6613472[a]	*1.6*
grey_16_16_10	242266[a]	*0.0*	**grey_16_16_41**	7002794[a]	*0.9*
grey_16_16_11	304722[a]	*0.1*	**grey_16_16_42**	7390586[a]	*1.5*

Instance name	Best known value	Time‡	Instance name	Best known value	Time‡
grey_16_16_12	368952 [a]	*0.1*	**grey_16_16_43**	7794422 [b]	*6.6*
grey_16_16_13	457504 [a]	*0.1*	**grey_16_16_44**	8217264 [b]	*18*
grey_16_16_14	547522 [a]	*0.1*	**grey_16_16_45**	8674910 [c]	*~130*
grey_16_16_15	644036 [a]	*0.1*	**grey_16_16_46**	9129192 [c]	*~90*
grey_16_16_16	742480 [a]	*0.1*	**grey_16_16_47**	9575736 [a]	*6.7*
grey_16_16_17	878888 [a]	*0.3*	**grey_16_16_48**	10016256 [a]	*4.1*
grey_16_16_18	1012990 [a]	*0.2*	**grey_16_16_49**	10518838 [b]	*7.1*
grey_16_16_19	1157992 [a]	*0.3*	**grey_16_16_50**	11017342 [a]	*8.9*
grey_16_16_20	1305744 [a]	*0.4*	**grey_16_16_51**	11516840 [b]	*12.5*
grey_16_16_21	1466210 [a]	*0.7*	**grey_16_16_52**	12018388 [b]	*11.9*
grey_16_16_22	1637794 [a]	*0.6*	**grey_16_16_53**	12558226 [a]	*14*
grey_16_16_23	1820052 [a]	*0.5*	**grey_16_16_54**	13096646 [b]	*8.9*
grey_16_16_24	2010846 [a]	*0.9*	**grey_16_16_55**	13661614 [b]	*19*
grey_16_16_25	2215714 [b]	*5.5*	**grey_16_16_56**	14229492 [b]	*5.8*
grey_16_16_26	2426298 [c]	*~50*	**grey_16_16_57**	14793682 [b]	*5.0*
grey_16_16_27	2645436 [a]	*1.6*	**grey_16_16_58**	15363628 [b]	*3.7*
grey_16_16_28	2871704 [a]	*1.7*	**grey_16_16_59**	15981086 [a]	*7.3*
grey_16_16_29	3122510 [a]	*1.4*	**grey_16_16_60**	16575644 [a]	*5.5*
grey_16_16_30	3373854 [a]	*0.9*	**grey_16_16_61**	17194812 [b]	*4.8*
grey_16_16_31	3646344 [a]	*1.2*	**grey_16_16_62**	17822806 [b]	*6.1*
grey_16_16_32	3899744 [a]	*0.9*	**grey_16_16_63**	18435790 [a]	*2.6*
grey_16_16_33	4230950 [a]	*1.3*	**grey_16_16_64**	19050432 [a]	*4.6*

‡ time (in seconds of 3GHz Pentium computer) needed to find the best known solution (BKS) under condition that all the 10 restarts out of 10 succeeded in finding the BKS;

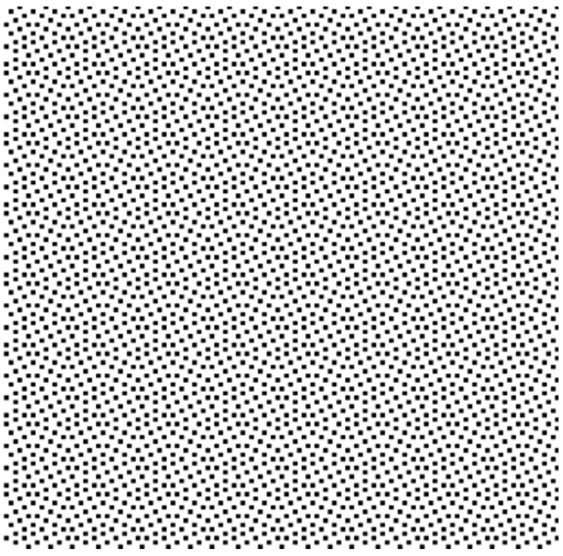
⋄ the optimality of these values has been proven by Drezner [6];

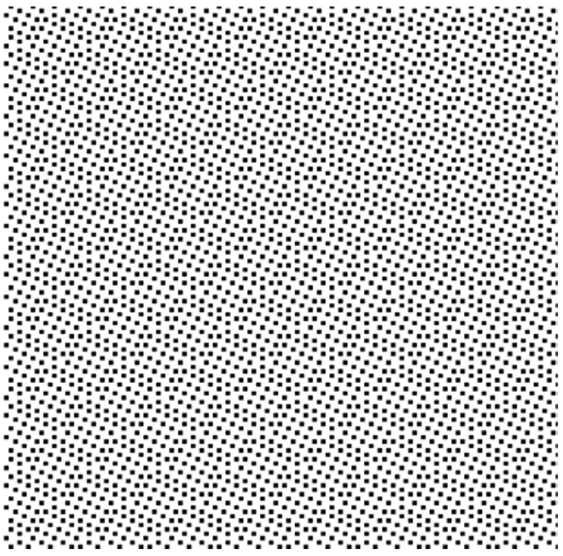
a reference: Taillard, Gambardella, 1997 [23]; b reference: Misevicius, 2003 [16]; c reference: Misevicius, 2003 [15].

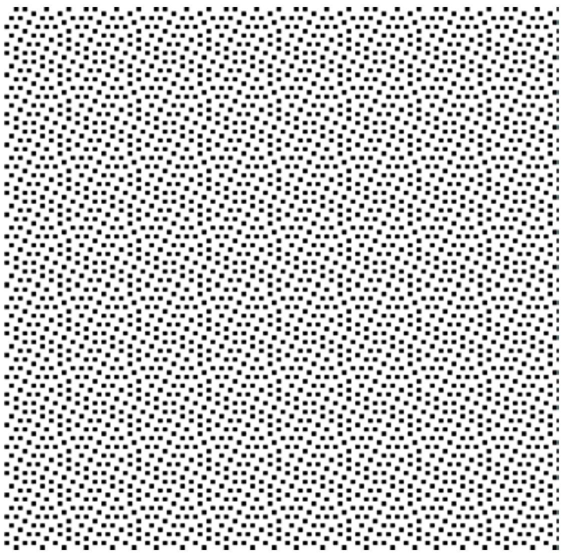
4. Concluding remarks

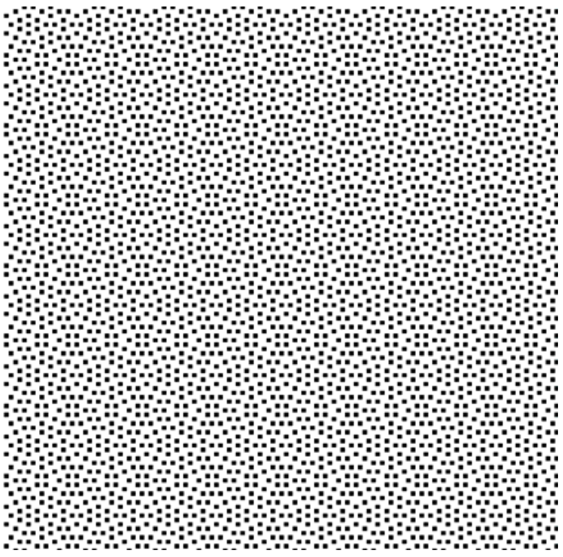
In this paper, we proposed a hybrid genetic algorithm that involves the innovative solution recombination operator – the so-called multiple parent crossover (MX). MX is distinguished for the important fact that the offspring derives the information from many parents – this is a quite contrast to the classical crossover operators where the inheritance of the information by the child is limited to two parents only. This original recombination operator coupled with other components of the hybrid genetic-tabu search resulted in a powerful optimization tool – the algorithm HGA-MX-TS. HGA-MX-TS was applied to the special case of the quadratic assignment problem, the grey pattern problem. The results obtained show promising performance of HGA-MX-TS. Sixty two instances of the grey pattern problem have been solved to pseudo-optimality at surprisingly short computational times with few exceptions. Some of these pseudo-optimal solutions were brought out in the graphical representation form.

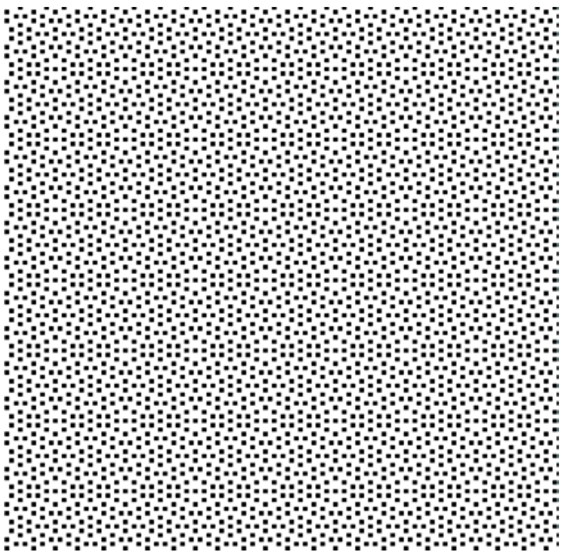
Our MX crossover is very aggressive and robust. Still, there is a room for the further enhancements. This is especially true due to the fact that some disadvantages of MX might be observed by performing more thorough experiments. One of the shortcomings is the loss of the diversity, especially, in the cases when the genetic search progresses and the individuals of the population tend to become very similar to each other. Consequently, if the degree of the diversity of the population is low, then the child produced by MX will, most likely, be just a copy of one of the parents. To overcome these (and other) difficulties, some improvements of the proposed MX are possible: a) incorporating the additional knowledge by constructing the desirability matrix, for example, the fitness (cost) of the individuals (solutions); b) adding noise (or some sort of fuzziness) to the desirability matrix; c) experimenting with the different numbers of the parents and/or population sizes. All these and, possibly, other extensions could be proper directions of the future research.


(a)


(b)


(c)


(d)


(e)

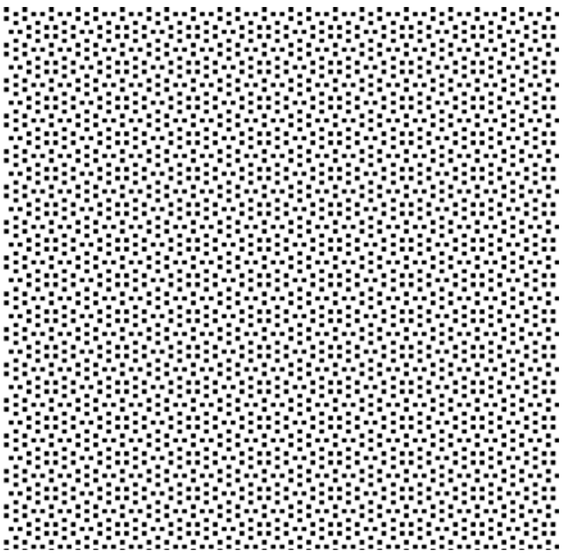
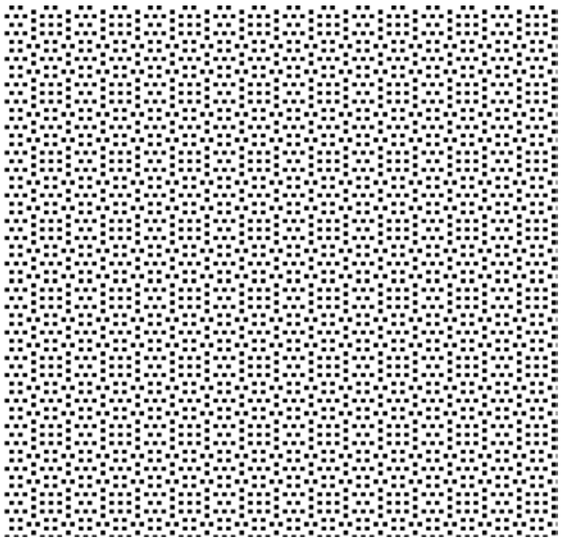
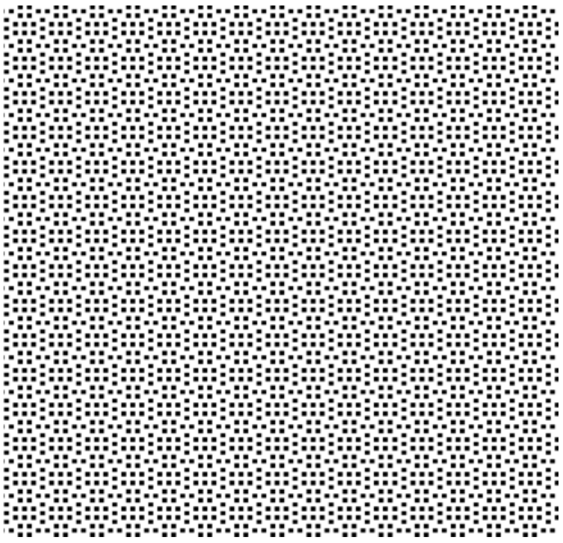
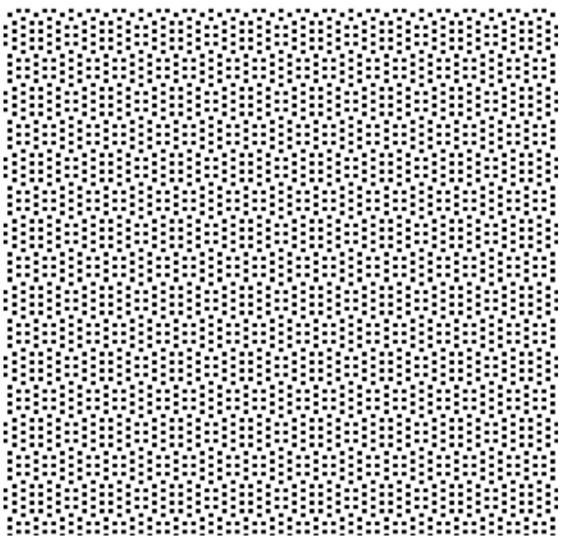
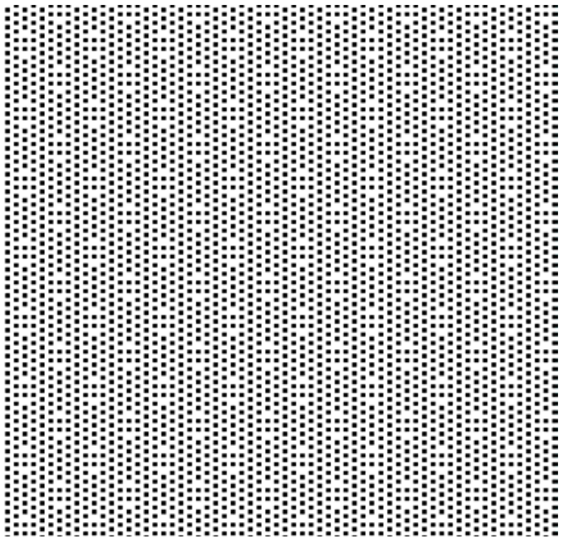

(f)

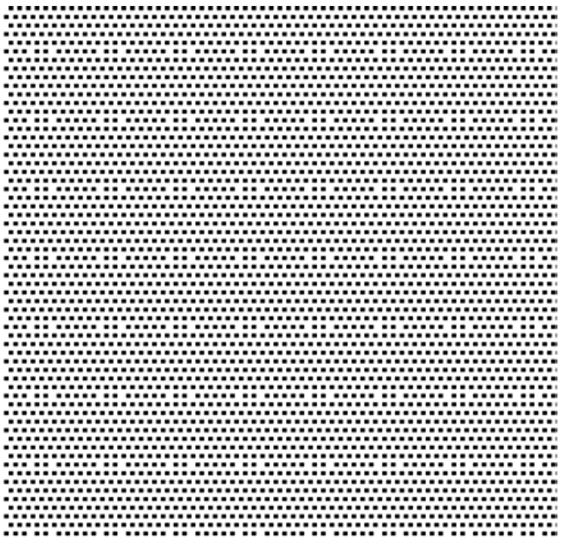
Figure 5. Examples of grey frames of densities 53/256 (a), 54/256 (b), 55/256 (c), 56/256 (d), 57/256 (e), 58/256 (f) (Part I)


(g)


(h)


(i)


(j)


(k)

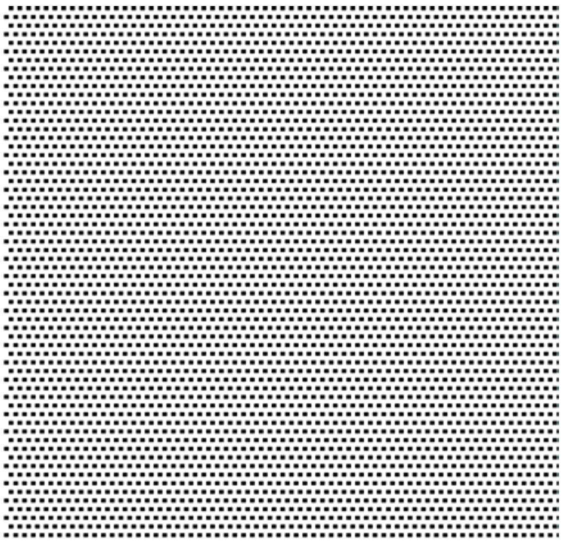

(l)

Figure 5. Examples of grey frames of densities 59/256 (g), 60/256 (h), 61/256 (i), 62/256 (j), 63/256 (k), 64/256 (l) (Part II)

References

[1] **R.K. Ahuja, J.B. Orlin, A. Tiwari.** A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 2000, *Vol.*27, 917–934.

[2] **R.E. Burkard, E. Çela, P.M. Pardalos, L. Pitsoulis.** The quadratic assignment problem. *In D.Z.Du, P.M. Pardalos (eds.), Handbook of Combinatorial Optimization, Kluwer, Dordrecht*, 1998, *Vol.*3, 241–337.

[3] **E. Çela.** The Quadratic Assignment Problem: Theory and Algorithms. *Kluwer, Dordrecht*, 1998.

[4] **L. Davis.** Handbook of Genetic Algorithms. *Van Nostrand, New York*, 1991.

[5] **Z. Drezner.** A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 2003, *Vol.*15, 320–330.

[6] **Z. Drezner.** Finding a cluster of points and the grey pattern quadratic assignment problem. *Working paper, College of Business and Economics, California State University-Fullerton, Fullerton, CA*, 2005.

[7] **C. Fleurent, F. Glover.** Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 1999, *Vol.*11, 198–204.

[8] **D.E. Goldberg.** Genetic Algorithms in Search, Optimization and Machine Learning. *Addison-Wesley, Reading*, 1989.

[9] **D.E. Goldberg, R. Lingle.** Alleles, loci, and the traveling salesman problem. *In J.J.Grefenstette (ed.), Proceedings of the First International Conference on Genetic Algorithms and their Applications*, *Lowrence Erlbaum, Hillsdale*, 1985, 154–159.

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

[11] **T. Koopmans, M. Beckmann.** Assignment problems and the location of economic activities. *Econometrica*, 1957, *Vol.*25, 53–76.

[12] **M.H. Lim, Y. Yuan, S. Omatu.** Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimization and Applications*, 2000, *Vol.*15, 249–268.

[13] **P. Merz, B. Freisleben.** A genetic local search approach for the quadratic assignment problem. *In T.Bäck (ed.), Proceedings of the Seventh International Conference on Genetic Algorithms*, *Morgan Kaufmann, East Lansing*, 1997, 465–472.

[14] **P. Merz, B. Freisleben.** Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 2000, *Vol.*4, 337–352.

[15] **A. Misevicius.** Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. *Knowledge-Based Systems*, 2003, *Vol.*16, 261–268.

[16] **A. Misevicius.** Ruin and recreate principle based approach for the quadratic assignment problem. *In E.Cantú-Paz, J.A.Foster, K.Deb et al. (eds.), Lecture Notes in Computer Science, Vol.*2723, *Genetic and Evolutionary Computation – GECCO* 2003, *Proceedings, Part* I, *Springer, Berlin-Heidelberg*, 2003, 598–609.

[17] **A. Misevicius.** An extension of hybrid genetic algorithm for the quadratic assignment problem. *Information Technology and Control*, 2004, *Vol.*4(33), 53–60.

[18] **A. Misevicius.** A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, 2005, *Vol.*30, 95–111.

[19] **P. Moscato.** Memetic algorithms: a short introduction. *In D.Corne, M.Dorigo, F.Glover (eds.), New Ideas in Optimization, McGraw-Hill, London*, 1999, 219–234.

[20] **I.M. Oliver, D.J. Smith, J.R.C. Holland.** A study of permutation crossover operators on the traveling salesman problem. *In J.J.Grefenstette (ed.), Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale*, 1987, 224–230.

[21] **C.R. Reeves, J.E. Rowe.** Genetic Algorithms: Principles and Perspectives. *Kluwer, Norwell*, 2001.

[22] **E. Taillard.** Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 1995, *Vol.*3, 87–105.

[23] **E. Taillard, L.M. Gambardella.** Adaptive memories for the quadratic assignment problem. *Tech. Report IDSIA*-87-97, *Lugano, Switzerland*, 1997.

[24] **D.M. Tate, A.E. Smith.** A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, 1995, *Vol.*1, 73–83.