INFORMATICA, 2006, Vol. 17, No. 2, 237–258

Experiments with Hybrid Genetic Algorithm for the Grey Pattern Problem

Alfonsas MISEVIČIUS
Department of Practical Informatics, Kaunas University of Technology
Studentų 50-400a/416a, LT-51368 Kaunas, Lithuania
e-mail: alfonsas.misevicius@ktu.lt

Received: July 2005

Abstract. Recently, genetic algorithms (GAs) and their hybrids have achieved great success in solving difficult combinatorial optimization problems. In this paper, the issues related to the performance of the genetic search in the context of the grey pattern problem (GPP) are discussed. The main attention is paid to the investigation of the solution recombination, i.e., crossover operators which play an important role by developing robust genetic algorithms. We implemented seven crossover operators within the hybrid genetic algorithm (HGA) framework, and carried out the computational experiments in order to test the influence of the recombination operators to the genetic search process. We examined the one point crossover, the uniform like crossover, the cycle crossover, the swap path crossover, and others. A so-called multiple parent crossover based on a special type of recombination of several solutions was tried, too. The results obtained from the experiments on the GPP test instances demonstrate promising efficiency of the swap path and multiple parent crossovers.

Key words: combinatorial optimization, heuristic algorithms, genetic algorithms, crossover operators, grey pattern problem, quadratic assignment problem.

1. Introduction

The grey pattern problem (GPP) (Taillard, 1995) 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 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) (Koopmans and Beckmann, 1957) 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 $\pi = (\pi(1), \pi(2), \ldots, \pi(n)) \in \Pi$ that minimizes

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

More complete discussions and surveys on the QAP can be found in (Burkard *et al.*, 1998; Çela, 1998; Pardalos and Resende, 2002; Pardalos and Wolkowicz, 1994; Pitsoulis and Pardalos, 2001; Rendl, 2002).

In the grey pattern problem, the matrix $(a_{ij})_{n\times n}$ is defined as $a_{ij} = 1$ for i, $j = 1, 2, \ldots, m$ 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

$$f_{rstu} = \max_{v,w\in\{-1,0,1\}} \frac{1}{(r-t+n_1v)^2+(s-u+n_2w)^2}, \qquad (2)$$
$$r,t = 1,\ldots,n_1, \quad s,u = 1,\ldots,n_2.$$

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, $j = 1, \ldots, n$) located in the positions k = $\pi(i)$ and l = $\pi(j)$ with the coordinates (r, s) and (t, u) (see also Taillard, 1995). The ith ($i \leqslant m$) element of the permutation (solution) π, $\pi(i) = n_2(r-1)+s$, gives the location in the rectangle where a black point has to be placed in. The coordinates of the location $\pi(i)$ of the black point are derived according to the formulas: $r = \lfloor (\pi(i)-1)/n_2 \rfloor + 1$, $s = ((\pi(i)-1) \bmod n_2) + 1$, $i = 1, 2, \ldots, m$.

Many heuristic approaches can be applied for solving both the QAP and, at that time, its particular case – the grey pattern problem (see, for example, Burkard *et al.*, 1998; Taillard, 1995). Recently, genetic algorithms (GAs) are among the advanced heuristic techniques for the combinatorial problems, like the QAP (see Ahuja *et al.*, 2000; Drezner, 2003; Merz and Freisleben, 2000; Misevicius, 2004b; Taillard and Gambardella, 1997).

Very roughly, genetic algorithms in the context of the QAP/GPP can be characterized as follows (Goldberg, 1989). Let P be a subset of Π; it is referred to as a population, and it is composed of individuals, i.e., solutions (permutations), $\pi_1, \pi_2, \ldots, \pi_{PS=|P|}$. Each individual ($\pi_i$) is associated with a fitness, i.e., the corresponding objective function value ($z(\pi_i)$). In this case, the individual π_i is preferred to individual π_j if $z(\pi_i) < z(\pi_j)$. (Further, we also shall call the solution (permutation), π, as a chromosome, the single position, i, of the solution (chromosome) – as a gene, and the value at the given position (gene), $\pi(i)$ – as an allele.) The following are the main steps of the genetic search. A pair (or fraction) of members of P is selected to be parents. New solutions (i.e., offspring) are created by combining (merging) the parents; this recombination operator is known as a crossover. Afterward, a replacement (culling) scheme is applied to the previous generation and the offspring to determine which individuals survive to form the next generation. In addition, some individuals may undergo mutations to prevent a premature loss of the diversity within the population. Over many generations, less fit individuals (worse solutions) tend to die-off, while better individuals (solutions) tend to predominate. The process is continued until a certain termination criterion is met.

For a more thorough discussion on the principles of GAs, the reader is addressed to (Davis, 1991; Goldberg, 1989; Reeves and Rowe, 2001).

In this paper, the issues related, namely to the genetic search for the grey pattern problem are discussed. The main attention is paid to the investigation of the solution

recombination (crossover) operators, which play an important role by constructing efficient genetic algorithms. The paper is organized as follows. A hybrid genetic algorithm (HGA) framework and recombination operators are discussed in Section 2. In Section 3, we present the results of testing of the various crossover operators within the HGA framework for the grey pattern problem. Section 4 completes the paper with conclusions.

2. A Hybrid Genetic Algorithm Framework and Recombination Operators

2.1. *A Hybrid Genetic Algorithm Framework: the State-of-the-Art and Further Extensions*

Traditional genetic algorithms face some difficulties, first of all, a huge amount of local optima over the search space. The second negative aspect is presence of cycles, i.e., repeating sequences of the search trajectories. There exists also the third phenomenon – the case in which the convergence to local optima and the cycles are absent but the search is still confined in limited portions of the solution space (this phenomenon is known as "chaotic attractors"). The algorithms that try to overcome these difficulties are rather hybrid, i.e., combined genetic local search algorithms which incorporate additional heuristic components (Moscato, 1999). The example of such component is a post-crossover procedure which is used to play the role of a local improvement algorithm applied to the solution previously produced by the crossover. Heuristic algorithms can also be applied for the construction of high quality initial populations. As a result, the genetic search is done in an optimized search space, where the populations consist solely of local optima – this appears to be much more effective process than when searching in a pure random solution space.

Applying hybrid genetic algorithms still does not necessarily mean that good solutions are reached at reasonable computation time. Indeed, HGAs often use the elaborated improvement heuristics (like simulated annealing, tabu search, etc.) that in general are quite time-consuming. This could be thought of as a serious shortcoming, especially if we wish to create HGAs that are competitive with other modern optimization techniques. In these circumstances, it is important to make some additional extensions to HGAs. The following are the basic principles by designing of the extended hybrid genetic algorithms (EHGAs).

1. EHGAs should incorporate as fast (robust) local improvement algorithms as possible. Here, we assume that the algorithm A_1 is "faster" than the algorithm A_2 (in the other words, A_1 dominates over A_2 in terms of run time) if A_1 finds (in average) the solution(s) with the average objective function value (quality) $f^{\diamondsuit}$ in less time (t_1) than A_2. Naturally, the long time behaviour does not matter as long as we are speaking of the fast algorithms in the context of EHGAs: the only matter is how quickly an algorithm achieves solution(s) with the quality $f^{\diamondsuit}$. One of the examples of robust algorithms is an iterated tabu search (ITS) (see, for example, Misevicius, 2005).

2. In EHGAs, the compactness of the population is greatly desirable. As long as powerful local improvement procedures are used, the large populations of solutions are not

necessary at all: the small size of the population is fully compensated by the robustness of the heuristic improvement algorithm. Obviously, the compact populations allow to save the computation time when comparing to other HGAs which deal with larger populations.

3. EHGAs must maintain a high degree of the diversity within the population. This is especially true for the small populations. Indeed, the smaller the size of the population, the larger the probability that the diversity will be lost quickly by using robust improvement heuristics. To fight this difficulty, so-called "cold restarts" may be proposed; here, as "cold restarts" we call deep reconstructions of the population, for example, the mutations applied to the members of a population with the subsequent local improvement (to keep the local optimality of the population). "Cold restart" takes place each time the fact of a premature convergence ("stagnation") is determined, i.e., the level of the diversity within the current population is below a certain threshold. As a measure for the diversity, entropy of the population may be used (Misevicius, 2003a).

The generalized framework of the extended hybrid genetic-tabu search algorithm is depicted in Fig. 1. Note that, within this framework, we applied a limited iterated tabu search procedure in the role of a local improvement algorithm. Remind that combining of the iterated tabu search and the genetic operators has been proven highly effective for the QAP (Misevicius, 2004a, 2004b). The favourable feature of the ITS procedure is that there is no need in the mutations within GA itself (which is the case in the ordinary GAs): each solution already undergoes mutation-like transformations during the execution of the iterated tabu search. The ITS procedure (but with the increased number of iterations) is also used at both the initial population construction and the restart process.

As stated above, the fast execution of the local improvement procedure is of high importance. This is even more true for the iterated tabu search where many iterations of the tabu search (TS) take place. Fortunately, as the GPP is a special case of the QAP,

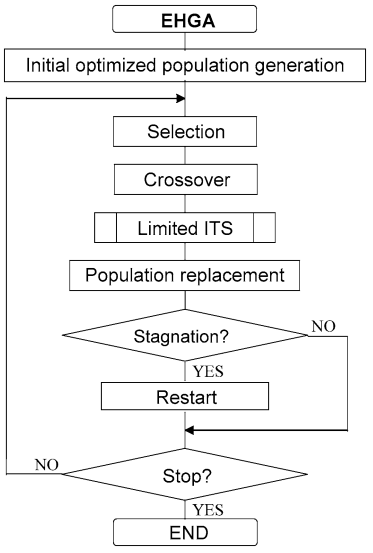

Fig. 1. Basic flowchart of the extended hybrid genetic algorithm.

lots of computations can be shorten and simplified due to the very specific character of the matrix $\boldsymbol{A}$ in the grey pattern problem, as shown in (Taillard, 1995). For the GPP, 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 $\mathrm{O}(m(n-m))$, instead of $\mathrm{O}(n^2)$ for the ordinary QAP. In addition, evaluating the difference in the objective function values becomes more faster because the matrix $\boldsymbol{A}$ is consisting of entries 0 and 1 only. So, instead of the standard formula of calculation difference in the values of the objective function when exchanging the ith and jth elements in the current permutation:

$$\begin{aligned}\Delta z(\pi,i,j) &= (a_{ii}-a_{jj})\big(b_{\pi(j)\pi(j)}-b_{\pi(i)\pi(i)}\big)+(a_{ij}-a_{ji})\big(b_{\pi(j)\pi(i)}-b_{\pi(i)\pi(j)}\big)\\ &+\sum_{k=1,k\neq i,j}^{n}(a_{ik}-a_{jk})\big(b_{\pi(j)\pi(k)}-b_{\pi(i)\pi(k)}\big)\\ &+\sum_{k=1,k\neq i,j}^{n}(a_{ki}-a_{kj})\big(b_{\pi(k)\pi(j)}-b_{\pi(k)\pi(i)}\big),\\ &\qquad i=1,2,\ldots,n-1,\quad j=i+1,\ldots,n,\end{aligned}\tag{3}$$

the simplified formula:

$$\begin{aligned}\Delta z(\pi,i,j) &= 2\sum_{k=1,k\neq i}^{m}\big(b_{\pi(j)\pi(k)}-b_{\pi(i)\pi(k)}\big),\\ &\qquad i=1,2,\ldots,m,\quad j=m+1,\ldots,n,\end{aligned}\tag{4}$$

is used. As a result, the TS algorithm complexity is reduced from $\mathrm{O}(n^3)$ to $\mathrm{O}(m^2(n-m))$.

Recently, Drezner (2005) proposed a very inventive trick which allows reducing the computation time even more. Based on this approach, $\Delta z(\pi,i,j)$ is calculated as follows:

$$\Delta z(\pi,i,j)=2\big(c_{\pi(j)}-c_{\pi(i)}-b_{\pi(i)\pi(j)}\big),\quad i=1,2,\ldots,m,\quad j=m+1,\ldots,n,\tag{5}$$

where $c_{\pi(i)}, c_{\pi(j)}$ are the entries of the array C of size n. The values of C are calculated once at the beginning of the algorithm for the initial solution π according to the formula $c_i=\sum_{j=1}^{m} b_{i\pi(j)}$, $i=1,2,\ldots,n$. Suppose that the permutation π' is obtained after the exchange of the kth and lth elements in the permutation π. Then, the (old) values of c_i are replaced by the following (new) values $c_i+b_{i\pi'(k)}-b_{i\pi'(l)}$, $i=1,2,\ldots,n$. This takes place after every $m(n-m)$ iterations only. The resulting complexity thus becomes $\mathrm{O}(mn)$. As the TS procedure is invoked many times during the execution of EHGA, the overall effect is even more evident, especially, in the cases when $m \ll n$.

The other details of the ITS algorithm are omitted for the sake of brevity. The reader interesting in ITS approach is addressed to (Misevicius, 2005).

2.2. *Recombination Operators*

HGAs incorporate local improvement procedures and operate with high quality optimized populations. Despite this fact, the recombination of solutions still remains one of the critical things by constructing competitive genetic algorithms. Very likely, the role of recombination operators within the hybrid genetic algorithms is more significant than in the canonical GAs. In fact, we can think of HGA as a process that combines intensification and diversification (I&D) of the search. The intensification (local improvement algorithm) concentrates the search in limited portions of the solution space, while the diversification is responsible for escaping from the current local optimum and moving towards unvisited so far solutions. (For more details on I&D methodology, see, for example, (Misevicius, 2003b, 2005).) From this point of view, the crossover is a special sort diversification (solution reconstruction) mechanism, which – generally speaking – guides the global search, i.e., exploration of new and new regions of the solution space. Hopefully, better locally optimal solutions will be discovered. Thus, the proper exploration strategy is, in some sense, even more severe than the intensification process, and may add a crucial influence on the resulting efficiency of the search.

2.2.1. *Basic Characteristics of Recombination Operators*

The crossover is one of the main genetic search operators. It is capable of producing a new feasible solution (i.e., child) by exchanging the information contained in both parents. From the philosophical point of view, crossover is a structured and, at that time, randomized process (operation) that guarantees both inheritance of the existing characteristics and creation of entirely new features. Mathematically, crossover can be defined as a binary operator (function) ψ: $\Pi \times \Pi \rightarrow \Pi$ such that $\psi(\pi', \pi'') \neq \pi' \vee \psi(\pi', \pi'') \neq \pi''$ if $\pi' \neq \pi''$; here, we assume that the solutions are represented by permutations. As a rule, the recombination operators ensure that the offspring definitely inherits the alleles which are common to both parents; more formally, $\pi'(i) = \pi''(i) \Rightarrow \pi^{\circ}(i) = \pi'(i) = \pi''(i)$, $i = 1, 2, \ldots, n$, where π', π'', π° are the parents and offspring, respectively. The inheritance of the remaining genes can be accomplished in a variety of ways. Before describing some of these ways, we discuss certain characteristics of the crossover operators, which seem to be quite severe in the context of HGAs. Firstly, the crossover operators can be characterized by a "distance" factor which may be viewed as a measure of how "far" is the offspring from the parents. Let $d(\pi^x, \pi^y)$ be Hamming distance between permutations π^x and π^y, i.e.:

$$d(\pi^x, \pi^y) = \Big|\big\{i \,|\, \pi^x(i) \neq \pi^y(i)\big\}\Big|. \tag{6}$$

The normalized integrated distance between the offspring and the parents, δ, is then defined as follows:

$$\delta = \frac{\delta_{\min} + \delta_{\max}}{2}, \tag{7}$$

where

$$\delta_{\min} = \frac{\min\{d(\pi^\circ, \pi'), d(\pi^\circ, \pi'')\}}{n}, \tag{8a}$$

$$\delta_{\max} = \frac{\max\{d(\pi^\circ, \pi'), d(\pi^\circ, \pi'')\}}{n}, \tag{8b}$$

where $d(\pi^\circ, \pi')$, $d(\pi^\circ, \pi'')$ are the distances between the offspring and the first and second parent, respectively. This normalized distance takes on values between 0 and 1. Of course, it is desirable that $\delta > 0$ to prevent the loss of the diversity and the premature convergence of GA.

The other important criterion of the crossover operators is a "degree of disruptiveness (randomness)". Disruptiveness can be thought of as a measure of "foreign" elements, i.e., elements that are not contained in the corresponding positions of the parents. More formally, the degree of disruptiveness, ρ, is defined as follows:

$$\rho = \frac{\left|\{i | \pi^\circ(i) \neq \pi'(i) \wedge \pi^\circ(i) \neq \pi''(i)\}\right|}{n}, \tag{9}$$

where π', π'' are the parents and π° is the offspring. It is obvious that $0 \leqslant \rho \leqslant 1$; in addition, $\delta_{\max} \geqslant \delta \geqslant \delta_{\min} \geqslant \rho$. The low degree of randomness does not necessarily mean that the distance is also small; however the experiments show some correlation between the statistical values of δ and ρ (see Section 3, Table 1). In general, there are two situations in the recombination process: 1) $\rho = 0; 2)\rho > 0$. The first situation is also referred to as an explicit mutation. Radcliffe and Surry (1994) use the term "assorting crossover". In this case, the offspring is different from both first and second parent (if $\delta_{\min} > 0$), in addition, every allele of the child is from the corresponding gene of either first or second parent; that is, $\pi^\circ \neq \pi' \wedge \pi^\circ \neq \pi'' \wedge (\pi^\circ(i) = \pi'(i) \vee \pi^\circ(i) = \pi''(i)$, $i = 1, 2, \ldots, n)$. This is a quite strict condition. That is the reason why the assorting recombination is hardly accomplished for some problems. The second situation is known as an implicit mutation (Radcliffe and Surry use the term "respectful crossover"). In this case, the only requirement to fulfil is that the offspring is different from the parents, i.e., $\delta_{\min} > 0$ (provided that the offspring necessarily inherits common parents' alleles). More formally, the implicit mutation takes place if there exists (at least one) such an i that $\pi^\circ(i) \neq \pi'(i) \wedge \pi^\circ(i) \neq \pi''(i)$.

The crossover operators can be classified as less disruptive (if ρ is relatively small) and more disruptive (if ρ is relatively large) (see Fig. 2). Within hybrid GAs, it could be conjectured that more disruptive crossovers are preferable to less disruptive crossovers. Indeed, as new solution produced by the recombination operator is again transformed into an optimized solution by the improvement heuristic, the crossover is highly desirable to be "strong" enough to allow to escape the current locally optimal solution and to move towards new regions in the solution space. Nevertheless, the crossover should also be "gentle" enough to keep the characteristics of the good solutions since parts of these solutions may be close to the ones of the global optimum. In addition, the following facts should be taken into consideration. If the recombination operator is too "disruptive", the

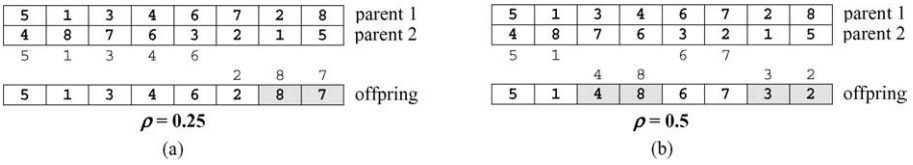

Fig. 2. Less disruptive crossover (a) vs more disruptive crossover (b).

resulting algorithm might be similar to a random multistart, which is known to be not a very efficient method. On the other hand, if the recombination operator is too "weak", the improvement procedure might possibly fall back into the previous local optimum.

2.2.2. *Conceptual Comparison of Recombination Operators*

Below we give a short outline (conceptual comparison) of several representatives of the recombination operators. We start our discussion with the one point crossover (OPX) operators (Goldberg, 1989). They are classical recombination procedures widely used in early versions of the genetic algorithms. One of the variants of OPX for the permutation-based solutions is due to Lim *et al.* (2000). The idea of OPX is quite simple. A crossing point (site) is chosen randomly between 1 and $n-1$ in one of the parents (say, π'). The corresponding alleles are copied to the offspring, the remaining elements are then copied from the "opposite" parent (π'') in such a way that the feasibility of the resulting solution is preserved. As a result, a child chromosome is obtained, containing information partially determined by each of parent chromosomes. The visual example of OPX is shown in Fig. 3.

The other popular crossover is based on a uniform recombination principle (Syswerda, 1989). The uniform like crossover (ULX) for the QAP was proposed by Tate and Smith (1995). ULX works as follows. First, all items assigned to the same position in both parents are copied to this position in the child. Second, the unassigned positions of a permutation are scanned from left to right: for the unassigned position, an item is chosen randomly, uniformly from those in the parents if they are not yet included in the child. Third, remaining items are assigned at random (see also Fig. 4).

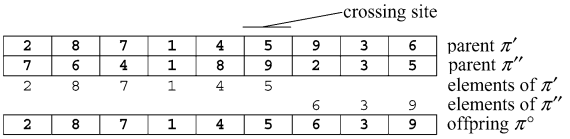

Fig. 3. Example of one point crossover.

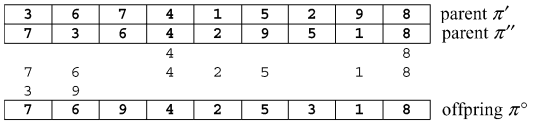

Fig. 4. Example of uniform like crossover.

Further, let us describe a variant of the well-known recombination operator, the partially-mapped crossover (PMX) (Goldberg and Lingle, 1985). The main idea of PMX is that it works with a part of a chromosome – mapping section – located between two crossover points (sites). PMX has been proven to be highly effective for the traveling salesman problem, however the straightforward PMX procedure does not work well for the QAP like problems. For this reason, Migkikh *et al.* (1996) proposed a modified partially-mapped crossover (MPMX) based on using a number of random mapping points – instead of one mapping segment. The basic steps of MPMX are as follows: a) clone the offspring π° from the first parent π'; b) choose a position pos_1 of the offspring at random; c) find a position pos_2 in the offspring where the content is equal to the content of pos_1 in the second parent π'', i.e., $\pi^\circ(pos_2) = \pi''(pos_1)$; d) swap the content of $\pi^\circ(pos_1)$ and $\pi^\circ(pos_2)$; e) repeat steps a-e k times, where $k = \lfloor \alpha n \rfloor$ (we used $\alpha = 0.15$). An illustration of MPMX is presented in Fig. 5.

The recombination operator we are going to outline falls into the category of "assorting" operators; it is quite different from those just discussed, which may be viewed as "respectful". This operator is known as a cycle crossover (CX) (Merz and Freisleben, 2000; Oliver *et al.*, 1987). The key point is that CX preserves the information contained in both parents, that is, all the alleles of the offspring are taken either from the first or second parent. The main steps of CX are as follows. 1. All the alleles found at the same locations in both parents are assigned to the corresponding locations in the child. 2 Starting from the first (or randomly chosen location) (provided that the corresponding element has not been included in the offspring yet), an element is chosen in a random way from the two parents. After this, one performs additional assignments to ensure that no random assignment (i.e., implicit mutation) occurs. Then, the next unassigned location is processed in the same manner until all the locations have been considered. An illustrative example is presented in Fig. 6.

Ahuja *et al.* (2000) implemented a swap path crossover (SPX) for the QAP, however the original idea of this type of recombination was developed by Glover (1994) under the entitlement "path relinking". So, let π', π'' be a pair of parents. In SPX, one starts at the first (or some random) gene, and the parents are examined from left to right until

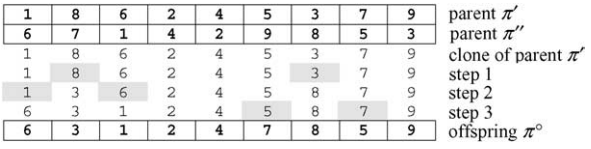

Fig. 5. Example of modified partially mapped crossover.

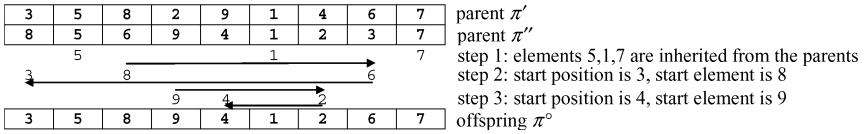

Fig. 6. Example of cycle crossover.

all the genes have been considered. If the alleles at the position being looked at are the same, one moves to the next position; otherwise, one performs a swap (interchange) of two alleles in π' or in π'' so that the alleles at the current position become alike. (For example, if the current gene is i, and $a = \pi'(i)$, $b = \pi''(i)$, then, after a swap, either $\pi'(i)$ becomes b, or $\pi''(i)$ becomes a.) Ahuja *et al.* (2000) suggest to perform the swap for which the corresponding solution has a lower cost (objective function value). The elements in the two resulting solutions are then considered, starting at the next position, and so on. The best solution obtained during this process (the fittest child) serves as an offspring. The "fragment" of the swap path crossover is illustrated in Fig. 7. The specific feature of SPX is that the problem-oriented knowledge (fitness of the parents/offspring) is taken into account. This is the contrast to the above crossovers, which may be viewed as "pure" operators.

Drezner (2003) introduced a quite interesting recombination operator – a cohesive crossover (COHX). The approach is based on maintaining a special distance matrix $\boldsymbol{M}$. The matrix $\boldsymbol{M}$ is filled in according to a wave propagation fashion starting from some initial (pivot) position (i_0, j_0) (see Fig. 8a). In case of the GPP, $\boldsymbol{M} = (m_{ij})_{n_1 \times n_2}$ where n_1, n_2 are the GPP dimensions. The matrix $\boldsymbol{M}$ corresponds to a one-dimensional vector $\boldsymbol{\mu}$ such that $m_{ij} = \mu_{(i-1)n_2+j}$, $i = 1, 2, \ldots, n_1$, $j = 1, 2, \ldots, n_2$ (see Fig. 8b).

Depending on the pivot position, there exist n different distance vectors: $\boldsymbol{\mu}^{(1)}$, $\boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(k)}, \ldots, \boldsymbol{\mu}^{(n)}$. Then, the kth recombined solution $\pi^{(k)}$ ($k = 1, 2, \ldots, n$) is generated in the following four steps:

1) the median, η, of $\boldsymbol{\mu}^{(k)}$ is calculated;

2) the positions which are closer than the median to the pivot position are assigned the alleles from the first (better) parent, i.e., $\pi^{(k)}(i) = \pi_{\text{better}}(i)$ if $\mu_i^{(k)} < \eta$, where $i = 1, 2, \ldots, n$, $\pi_{\text{better}} = \text{argmin}\,\{z(\pi'), z(\pi'')\}$, π', π'' are the solutions-parents;

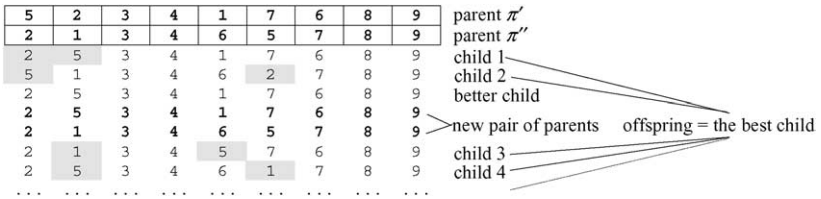

Fig. 7. Example of swap path crossover.

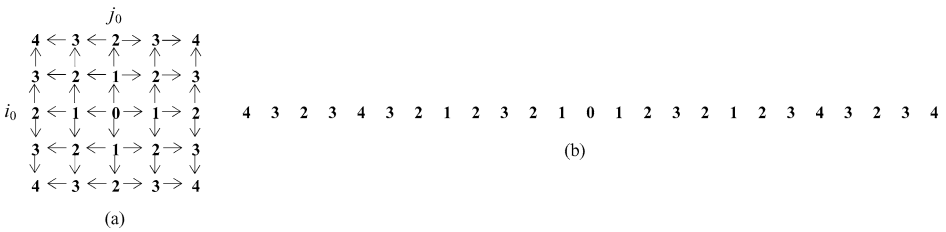

Fig. 8. Example of the special distance matrix: two-dimensional (a) and one-dimensional (b) representations.

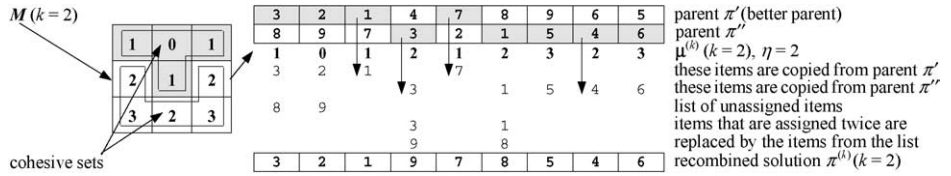

Fig. 9. Example of cohesive crossover.

3) all other positions are assigned the alleles from the second (worse) parent, i.e., $\pi^{(k)}(i) = \pi_{\text{worse}}(i)$ if $\mu_i^{(k)} \geqslant \eta$, where $i = 1, 2, \ldots, n$, $\pi_{\text{worse}} = \text{argmax}\,\{z(\pi'), z(\pi'')\}$;

4) it is possible that some alleles are assigned twice and some are not assigned at all; so, a list of unassigned alleles is created and all alleles from the second parent that are assigned twice are replaced with an allele from the list (provided that the allele is not yet included in the offspring).

An example of generation of the recombined solution is given in Fig. 9. In all, n solutions are produced, but only the best of them is regarded as an offspring, i.e., $\pi^\circ = \arg\min_{k=1,2,\ldots n} z(\pi^{(k)})$. It should be noted that the vectors $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(n)}$ may simply be substituted by the "real distances", i.e., the corresponding rows of the matrix $\boldsymbol{B}$ (see Section 1). This results in an effective problem-oriented crossover. For more details on COHX, see (Drezner, 2003).

Multiple parent crossover (MPX) was described by Misevicius and Rubliauskas (2005), although the idea of using combinations of several solutions goes back to (Boese *et al.*, 1994; Fleurent and Glover, 1999; Mühlenbein, 1989). MPX is distinguished for the fact that the offspring derives the information from many parents – this is the contrast and, at that time, the advantage to the traditional operators, where the useful information may be left out of account because of using two parents only. In MPX, ith element, i.e., allele of the offspring π° is created by choosing such a number j (among those not yet chosen) that the probability that $\pi^\circ(i) = j$ $\Pr(\pi^\circ(i) = j)$ is maximized. Here, the probability $\Pr(\pi^\circ(i) = j)$ is equal to $d_{ij}/\sum_j^n d_{ij}$, where d_{ij} is the entry of a so-called desirability matrix $\boldsymbol{D} = (d_{ij})_{n\times n}$; the value of d_{ij} is determined by the sum $q_{ij} + \varepsilon$, where q_{ij} is the number of times that the element i is assigned to the position $j = \pi(i)$ in μ parents (which participate in creation of the child), and ε is a correction (noise). The process is to be continued until all the genes of the offspring take on their values. The example of producing of the offspring in multiple parent crossover ($\mu = 5$) is given in Fig. 10.

3. Testing of the Hybrid Genetic Algorithm for the Grey Pattern Problem

In this section, we present the results of the experimental comparison of the crossovers outlined above. In the experiments, we used the instances of the GPP generated according to the method described in (Taillard, 1995). For the set of problems tested, the size of the instances, n, is equal to 256, and the frames (rectangles) are of dimensions 16×16, i.e., $n_1 = n_2 = 16$. The instances are denoted by the name grey_16_16_*m*, where m is the density of grey; it varies from 3 to 128. Remind that, for these instances, the data

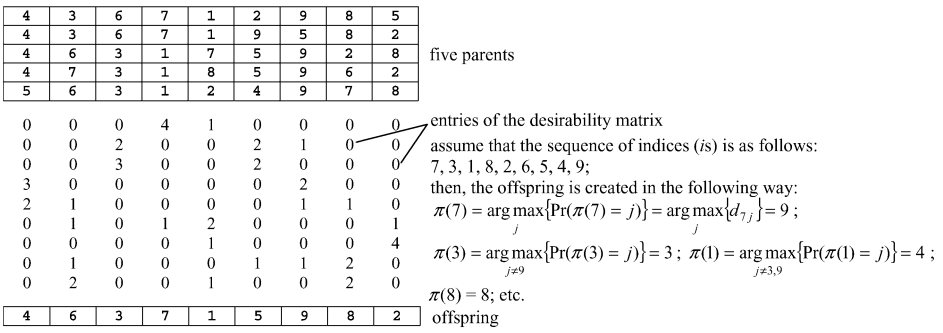

Fig. 10. Example of multiple parent crossover.

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 sub-matrix of size $m \times m$ composed of 1s only (Taillard and Gambardella, 1997).

Firstly, we conducted a small experiment to determine the approximate (empirical) values of the characteristics discussed in Section 2.2.1 for the two-parent crossovers. These values are presented in Table 1. They confirm that SPX, CX are less disruptive, while ULX, OPX are more disruptive, as conjectured. The only exception was the relatively high degree of disruption for COHX. For the respectful crossovers (OPX, ULX, MPMX, SPX, COHX), we found that the average values of δ (the integrated distance), ρ (the degree of disruptiveness), and the aspect ratio $\frac{\delta}{\rho}$ are approximately equal to **0.5**, **0.13**, and **4**, respectively.

We then carried out more thorough computational experiments. The goal was to find out how difficult are the grey pattern problems for different recombination operators. As an experimental basis for the crossovers, we used the extended hybrid genetic-tabu search algorithm discussed in Section 2.1. The efficiency measure for the crossovers is the average deviation of the solutions obtained from the best known solution – AD ($AD = 100(\bar{z} - \check{z})/\check{z}[\%]$, where $\bar{z}$ is the average objective function value over 10 restarts (single applications of EHGA to a given instance), and $\check{z}$ is the best known value (BKV) of the objective function).

Table 1

The empirical values of the characteristics of the crossovers

Factors	**Crossovers**					
	OPX	**ULX**	**MPMX**	**CX**	**SPX**	**COHX**
$\boldsymbol{\delta}_{\min}$	0.27	0.55	0.25	0.16	0.025	0.35
$\boldsymbol{\delta}_{\max}$	0.69	0.64	0.72	0.66	0.95	0.54
$\boldsymbol{\delta}$	0.48	0.59	0.49	0.41	0.49	0.44
$\boldsymbol{\rho}$	0.13	0.23	0.12	0	0.012	0.16

Note. The values of $\delta_{\min}, \delta_{\max}, \delta$, and ρ were obtained on the GPP instance grey_16_16_50 by performing 100 calls to the corresponding crossover procedure.

In the experimental comparison, equated conditions are created: all the crossover variants use the identical initial solutions and require approximately the same CPU time (some fluctuations in CPU times are due to the non-deterministic number of restarts during the execution of EHGA). The following are the values of the control parameters of EHGA (of course, they are equivalent for all the crossovers compared): population size -8; number of generations -25; number of offspring (crossovers) per generation -1; number of iterations of the post-crossover, i.e., the iterated tabu search procedure $-\frac{1}{10}n^2$.

Table 2

Results of the experiments with the GPP I: comparison of the crossover operators

Instance	BKV	*AD*							CPU time
		OPX	ULX	MPMX	CX	SPX	COHX	MPX	
grey_16_16_10	242266[a]	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*1.5*
grey_16_16_15	644036[a]	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*1.7*
grey_16_16_20	1305744[a]	**0.000**	0.007	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*1.9*
grey_16_16_25	2215714[b]	0.012	0.011	**0.004**	0.005	0.008	0.009	0.011	*2.2*
grey_16_16_30	3373854[a]	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*2.5*
grey_16_16_35	4890132[a]	0.011	0.005	**0.001**	0.007	0.008	0.012	0.005	*2.7*
grey_16_16_40	6613472[a]	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*3.0*
grey_16_16_45	8674910[c]	0.069	0.110	0.062	0.061	**0.041**	0.047	0.048	*3.2*
grey_16_16_50	11017342[a]	0.005	0.011	0.003	0.004	**0.001**	**0.001**	**0.001**	*3.5*
grey_16_16_55	13661614[b]	0.019	0.060	0.025	0.024	**0.016**	0.021	0.017	*5.0*
grey_16_16_60	16575644[a]	0.004	**0.000**	**0.000**	0.002	**0.000**	0.008	**0.000**	*5.9*
grey_16_16_65	19848790[b]	0.022	0.023	0.019	**0.000**	0.012	0.025	0.006	*6.5*
grey_16_16_70	23852796[b]	0.176	0.217	0.171	0.176	0.167	0.164	**0.140**	*6.8*
grey_16_16_75	28114952[b]	0.058	0.111	0.078	0.058	0.057	0.053	**0.048**	*7.0*
grey_16_16_80	32593088[b]	0.095	0.147	0.122	0.106	0.095	0.077	**0.045**	*7.2*
grey_16_16_85	37379304[b]	0.053	0.114	0.055	0.055	**0.029**	0.046	0.039	*7.7*
grey_16_16_90	42608826[c]	0.057	0.098	0.063	0.067	0.044	0.042	**0.037**	*8.3*
grey_16_16_95	48081112[d]	0.129	0.167	0.149	0.136	0.123	0.123	**0.122**	*8.5*
grey_16_16_100	53838088[a]	0.090	0.111	0.104	0.100	**0.081**	0.092	0.090	*9.0*
grey_16_16_105	59854744[a]	**0.085**	0.091	0.089	0.102	0.092	0.092	0.097	*9.7*
grey_16_16_110	66120434[d]	0.088	0.057	0.071	0.051	**0.044**	0.071	0.083	*10.2*
grey_16_16_115	72630764[a]	0.019	0.073	0.024	0.031	**0.000**	**0.000**	**0.000**	*11.1*
grey_16_16_120	79375832[a]	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*12.3*
grey_16_16_125	86327812[a]	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*13.4*
Average:		0.040	0.057	0.042	0.039	**0.033**	**0.035**	**0.032**	

Notes. 1. The best results obtained are printed in bold face.
2. Average CPU times per restart are given in seconds.
3. 3 GHz PENTIUM computer was used in the experiments.

[a] reference: (Taillard and Gambardella, 1997);
[b] reference: (Misevicius, 2003b);
[c] reference: (Misevicius, 2003a);
[d] reference: (Misevicius, 2005).

The number of parents in the MPX crossover is equal to the population size. The results of comparison are presented in Table 2.

The deviation averaged over the set of instances examined (see the last row of Table 2) may not reflect all aspects of the performance of crossovers. For example, the performance of MPMX is rather poor with respect to AD accumulated over 24 instances, but we can see that, by using this operator, the number of times that all 10 restarts out of 10 succeed in finding BKV, i.e., $AD = 0$, is greater or equal than when using the OPX, ULX, CX, and COHX operators. Our analysis is therefore based on several cumulative measures. These measures are collected in Table 3.

The results from Tables 2, 3 are quite "flat" and the differences in values of the performance measures are rather small. Most probably, this is due to the powerful post-crossover procedure used; on the other hand, this may have be caused by the specific nature of the GPP instances, which look to be easy, in particular, for the extended hybrid genetic algorithm. Nevertheless, it can be seen that the crossover operators, which are responsible for the exploration of new regions in the solution space, hide certain potential and still have appropriate influence on the final solutions.

The regularities we observed are as follows. It seems, with few exceptions, that less disruptive crossovers (for example, SPX) tends to be more efficient than the crossovers with a higher degree of disruption (for example, ULX); but the cycle crossover (the minimally available disruptive crossover) produces only medium-quality results. So, it could be conjectured that a good crossover should bring some randomness to the offspring, however this must be done in a subtle way. It can also be noted that the crossovers that incorporate some a priori knowledge about the problem being solved (for example, COHX, SPX) appear to be better than the "pure" operators (for example, OPX, ULX). Lastly, the ordinary two-parent crossovers are somewhat inferior to the multiple parent crossover,

Table 3

Cumulative measures and ranking of the crossovers

Cumulative measures	**Values (ranks)**						
	OPX	**ULX**	**MPMX**	**CX**	**SPX**	**COHX**	**MPX**
CUMAD	0.040 (5)	0.057 (7)	0.042 (6)	0.039 (4)	0.033 (2)	0.035 (3)	0.032 (1)
MINAD	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MAXAD	0.176 (5-6)	0.217 (7)	0.171 (4)	0.176 (5-6)	0.167 (3)	0.164 (2)	0.140 (1)
MEDIAN	0.019 (5)	0.040 (7)	0.022 (6)	0.016 (3)	0.010 (2)	0.017 (4)	0.009 (1)
$N_{AD=0}$	7 (6-7)	7 (6-7)	8 (3-5)	8 (3-5)	9 (1-2)	8 (3-5)	9 (1-2)
Resulting rank:	**6**	**7**	**5**	**4**	**2**	**3**	**1**

The following notations are used:
CUMAD – cumulative average deviation (over 24 instances);
MINAD – minimum average deviation;
MAXAD – maximum average deviation;
MEDIAN – median of deviations;
$N_{AD=0}$ – number of times that all 10 restarts out of 10 succeeded in finding BKV

which is distinguished for the fact that many individuals take part in producing an offspring (i.e., "orgies" take place). However, we found that MPX consumes some more CPU time.

We were also interested in seeing how our hybrid genetic-tabu search algorithm compares to other approaches. With that end in view, we provide some additional results of comparison of EHGA with other three genetic algorithm variants: 1) GA1 (Tate and Smith, 1995); 2) GA2 (Lim *et al.*, 2000); 3) GA3 (Fleurent and Ferland, 1994). GA1 is a simple genetic algorithm without local improvement (except the optimized initial population). The next two GA versions use local optimization procedures (descent local search – in GA2, tabu search – in GA3); however, the remaining structure of these algorithms is quite different from that of EHGA. The results obtained on 11 selected GPP instances (see Table 4 and Fig. 11) confirm the power of EHGA, which incorporates both the rapid (robust) procedure for local improvement and the right mechanism for avoiding premature convergence.

The goal of the further extensive experimentation was to determine the average computation time needed to find the pseudo-optimal (best know) solutions for the GPP instances grey_16_16_3..grey_16_16_128. For each of these instances, we performed 30 independent runs (multi-restarts) each consisting of 10 restarts of EHGA. The SPX operator (ranked as the second, but slightly faster than MPX) was utilized within EHGA at these long runs. Various combinations of the control parameters were used in the different runs. The best CPU times obtained during these multi-restarts are presented in Table 5. The corresponding graphical illustration is depicted in Fig. 12.

Table 4
Results of comparison of EHGA with other algorithms on 11 GPP instances

Instance	***AD***				**CPU time**
	GA1	**GA2**	**GA3**	**EHGA**	
grey_16_16_50	3.273	0.884	0.231	**0.001**	*3.5*
grey_16_16_55	4.446	0.817	0.200	**0.017**	*5.0*
grey_16_16_60	5.682	1.418	0.103	**0.000**	*5.9*
grey_16_16_65	6.373	2.360	0.123	**0.006**	*6.5*
grey_16_16_70	5.303	1.301	0.328	**0.140**	*6.8*
grey_16_16_75	4.447	0.844	0.207	**0.048**	*7.0*
grey_16_16_80	4.342	0.754	0.265	**0.045**	*7.2*
grey_16_16_85	4.605	0.769	0.207	**0.039**	*7.7*
grey_16_16_90	4.305	0.552	0.177	**0.037**	*8.3*
grey_16_16_95	4.202	0.549	0.247	**0.122**	*8.5*
grey_16_16_100	4.196	0.510	0.202	**0.090**	*9.0*
Average:	4.652	0.978	0.208	**0.050**	

Note. The MPX operator was utilized within EHGA

Table 5
Results of the experiments with the GPP II: CPU times needed to find pseudo-optimal solutions

Instance	Best known value	Time‡	Instance	Best known value	Time‡	Instance	Best known value	Time‡
grey_16_16_3	7810[a]	*0.0*	**grey_16_16_45**	8674910[c]	*180*	**grey_16_16_87**	39389054[b]	*26.1*
grey_16_16_4	15620[a]	*0.0*	**grey_16_16_46**	9129192[c]	*74*	**grey_16_16_88**	40416536[b]	*25.6*
grey_16_16_5	38072[a]	*0.0*	**grey_16_16_47**	9575736[a]	*3.4*	**grey_16_16_89**	41512742[b]	*211*
grey_16_16_6	63508[a]	*0.0*	**grey_16_16_48**	10016256[a]	*2.1*	**grey_16_16_90**	42597626[d]	*198*
grey_16_16_7	97178[a]	*0.0*	**grey_16_16_49**	10518838[b]	*3.6*	**grey_16_16_91**	43676474[d]	*275*
grey_16_16_8	131240[a]	*0.0*	**grey_16_16_50**	11017342[a]	*2.9*	**grey_16_16_92**	44759294[f]	*200*
grey_16_16_9	183744[a]	*0.0*	**grey_16_16_51**	11516840[b]	*7.8*	**grey_16_16_93**	45870244[d]	*297*
grey_16_16_10	242266[a]	*0.0*	**grey_16_16_52**	12018388[b]	*7.3*	**grey_16_16_94**	46975856[d]	*250*
grey_16_16_11	304722[a]	*0.1*	**grey_16_16_53**	12558226[a]	*8.6*	**grey_16_16_95**	48081112[g]	*231*
grey_16_16_12	368952[a]	*0.1*	**grey_16_16_54**	13096646[b]	*4.5*	**grey_16_16_96**	49182368[a]	*297*
grey_16_16_13	457504[a]	*0.1*	**grey_16_16_55**	13661614[b]	*11.7*	**grey_16_16_97**	50344050[a]	*327*
grey_16_16_14	547522[a]	*0.1*	**grey_16_16_56**	14229492[b]	*3.0*	**grey_16_16_98**	51486642[a]	*223*
grey_16_16_15	644036[a]	*0.1*	**grey_16_16_57**	14793682[b]	*2.5*	**grey_16_16_99**	52660116[a]	*236*
grey_16_16_16	742480[a]	*0.1*	**grey_16_16_58**	15363628[b]	*2.4*	**grey_16_16_100**	53838088[a]	*132*
grey_16_16_17	878888[a]	*0.2*	**grey_16_16_59**	15981086[a]	*3.7*	**grey_16_16_101**	55014262[a]	*98*
grey_16_16_18	1012990[a]	*0.1*	**grey_16_16_60**	16575644[a]	*2.8*	**grey_16_16_102**	56202826[g]	*47*
grey_16_16_19	1157992[a]	*0.2*	**grey_16_16_61**	17194812[b]	*2.4*	**grey_16_16_103**	57417112[a]	*81*
grey_16_16_20	1305744[a]	*0.3*	**grey_16_16_62**	17822806[b]	*3.6*	**grey_16_16_104**	58625240[g]	*78*
grey_16_16_21	1466210[a]	*0.5*	**grey_16_16_63**	18435790[a]	*1.9*	**grey_16_16_105**	59854744[a]	*44.5*
grey_16_16_22	1637794[a]	*0.3*	**grey_16_16_64**	19050432[a]	*2.3*	**grey_16_16_106**	61084902[a]	*40.2*
grey_16_16_23	1820052[a]	*0.2*	**grey_16_16_65**	19848790[b]	*3.2*	**grey_16_16_107**	62324634[a]	*23.3*
grey_16_16_24	2010846[a]	*0.6*	**grey_16_16_66**	20648754[b]	*4.7*	**grey_16_16_108**	63582416[a]	*13.3*
grey_16_16_25	2215714[b]	*3.2*	**grey_16_16_67**	21439396[b]	*11.5*	**grey_16_16_109**	64851966[a]	*14.4*
grey_16_16_26	2426298[c]	*18.5*	**grey_16_16_68**	22234020[b]	*22.0*	**grey_16_16_110**	66120434[g]	*13.8*
grey_16_16_27	2645436[a]	*1.1*	**grey_16_16_69**	23049732[b]	*31.5*	**grey_16_16_111**	67392724[a]	*9.7*
grey_16_16_28	2871704[a]	*0.9*	**grey_16_16_70**	23852796[b]	*33.7*	**grey_16_16_112**	68666416[a]	*8.8*
grey_16_16_29	3122510[a]	*0.8*	**grey_16_16_71**	24693608[b]	*86*	**grey_16_16_113**	69984758[a]	*11.2*
grey_16_16_30	3373854[a]	*0.5*	**grey_16_16_72**	25529984[b]	*97*	**grey_16_16_114**	71304194[a]	*7.5*
grey_16_16_31	3646344[a]	*0.7*	**grey_16_16_73**	26375828[d]	*345*	**grey_16_16_115**	72630764[a]	*5.6*
grey_16_16_32	3899744[a]	*0.6*	**grey_16_16_74**	27235240[e]	*350*	**grey_16_16_116**	73962220[a]	*5.8*
grey_16_16_33	4230950[a]	*0.7*	**grey_16_16_75**	28114952[b]	*53*	**grey_16_16_117**	75307424[a]	*4.1*
grey_16_16_34	4560162[a]	*2.7*	**grey_16_16_76**	29000908[b]	*134*	**grey_16_16_118**	76657014[a]	*3.9*
grey_16_16_35	4890132[a]	*3.3*	**grey_16_16_77**	29894452[e]	*172*	**grey_16_16_119**	78015914[a]	*2.5*
grey_16_16_36	5222296[a]	*2.0*	**grey_16_16_78**	30797954[e]	*129*	**grey_16_16_120**	79375832[a]	*1.9*
grey_16_16_37	5565236[a]	*1.8*	**grey_16_16_79**	31702182[b]	*14.2*	**grey_16_16_121**	80756852[a]	*1.7*
grey_16_16_38	5909202[a]	*0.9*	**grey_16_16_80**	32593088[b]	*3.8*	**grey_16_16_122**	82138768[a]	*1.5*
grey_16_16_39	6262248[a]	*1.1*	**grey_16_16_81**	33544628[b]	*4.4*	**grey_16_16_123**	83528554[a]	*1.0*
grey_16_16_40	6613472[a]	*0.9*	**grey_16_16_82**	34492592[b]	*81*	**grey_16_16_124**	84920540[a]	*0.7*
grey_16_16_41	7002794[a]	*0.6*	**grey_16_16_83**	35443938[d]	*65*	**grey_16_16_125**	86327812[a]	*0.4*
grey_16_16_42	7390586[a]	*0.7*	**grey_16_16_84**	36395172[d]	*71*	**grey_16_16_126**	87736646[a]	*0.3*
grey_16_16_43	7794422[b]	*3.4*	**grey_16_16_85**	37378800[d]	*186*	**grey_16_16_127**	89150166[a]	*0.3*
grey_16_16_44	8217264[b]	*17.2*	**grey_16_16_86**	38376438[c]	*125*	**grey_16_16_128**	90565248[a]	*0.2*

‡ time is given in seconds (on 3GHz Pentium computer) that is needed to find the best known solution (BKS) under condition that all the 10 restarts out of 10 succeeded in finding BKS;
[a] reference: (Taillard and Gambardella, 1997); [b] reference: (Misevicius, 2003b); [c] reference: (Misevicius, 2003a);
[d] reference: this paper; [e] reference: (Misevicius, 2004b); [f] reference: (Stützle, 1997);
[g] reference: (Misevicius, 2005).

There are several distinct "regions" of densities m (see Fig. 12). The instances with $m \leqslant 44$, $m = 47..70$, $m = 80, 81$, $m = 87, 88$, $m \geqslant 105$ are very easy to solve by the extended hybrid genetic algorithm. The instances with the remaining values of m appear to be relatively difficult for EHGA. Nevertheless, the overall performance of EHGA is

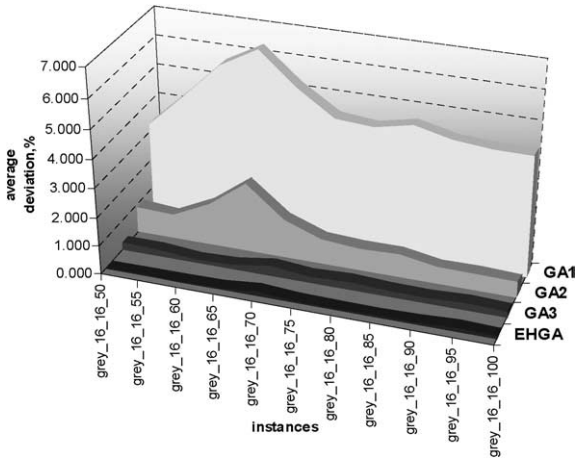

Fig. 11. Results of comparison of EHGA with other algorithms.

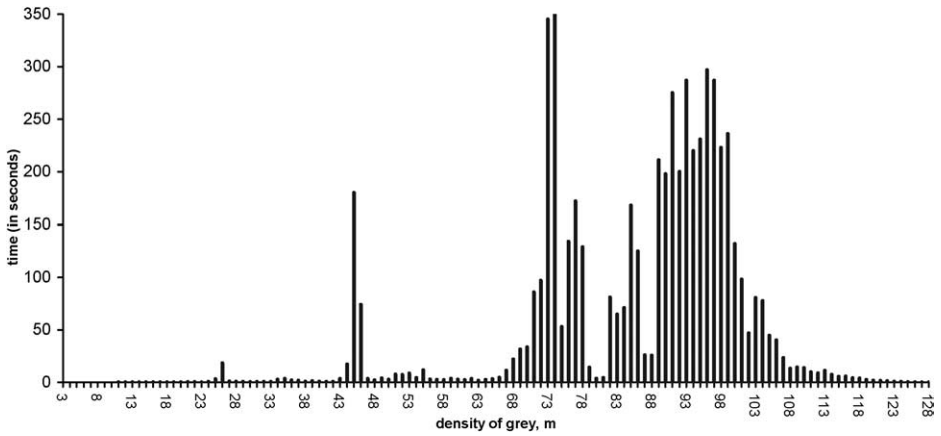

Fig. 12. Illustration of CPU times needed to find pseudo-optimal solutions for the GPP instances.

really promising. We guess that the run times may be decreased even more by a more careful tuning of the control parameters of EHGA.

During the experimentation, we were successful in finding new record-breaking solutions for eight GPP instances with $m = 73, 83, 84, 85, 90, 91, 93, 94$[1]. These solutions are presented in Table 6. As a confirmation of the quality of new solutions obtained, we also give the visual representation of these solutions in Fig. 13 – the reader can thus judge the quality of the grey frames obtained.

[1]During the preparation of our paper we found out that, simultanesly, Drezner (2005) obtained the same new solutions; he also proved the optimality of the solutions for the instances grey_16_16_3..grey_16_16_8 by using a branch and bound algorithm.

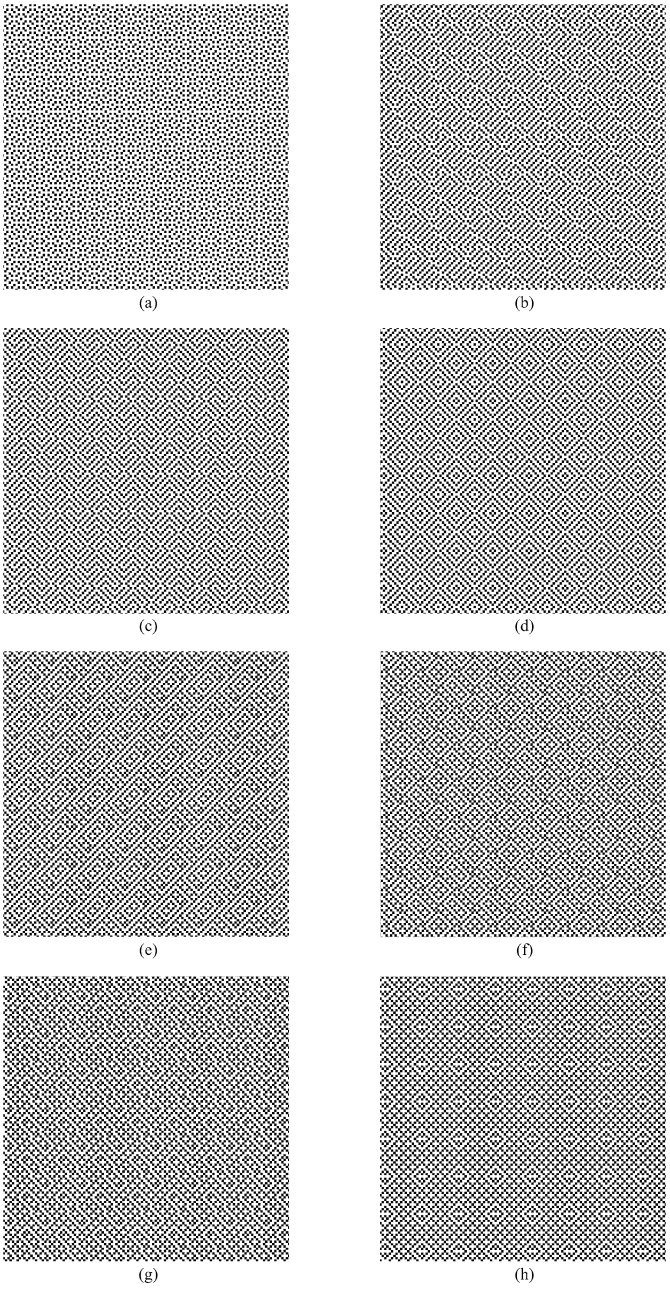

Fig. 13. Pseudo-optimal grey frames of densities 73/256 (a), 83/256 (b), 84/256 (c), 85/256 (d), 90/256 (e), 91/256 (f), 93/256 (g), 94/256 (h).

Table 6
New best known solutions for the GPP

Instance	Previous best known value	New best known value
grey_16_16_73	26382310[a]	26375828
grey_16_16_83	35444806[b]	35443938
grey_16_16_84	36397376[b]	36395172
grey_16_16_85	37379304[a]	37378800
grey_16_16_90	42608826[b]	42597626
grey_16_16_91	43694968[c]	43676474
grey_16_16_93	45883642[c]	45870244
grey_16_16_94	46979436[c]	46975856

[a]reference: (Misevicius, 2003b);
[b]reference: (Misevicius, 2003a);
[c]reference: (Misevicius, 2005).

4. Conclusions

In this paper, we presented the results of the experiments of the effective hybrid genetic algorithm when applied to the grey pattern problem, the special case of the quadratic assignment problem, which is known to be NP-hard. The main attention was paid to the investigation of the crossover operators which play one of the main roles in the constructing of the competitive genetic algorithms.

We implemented seven different crossover procedures within the extended hybrid genetic algorithm framework and carried out the extensive experiments in order to find out how is the difference of the quality of solutions produced by the different crossovers. We examined the one point crossover (OPX), the uniform like crossover (ULX), the modified partially-mapped crossover (MPMX), the cycle crossover (CX), the swap path crossover (SPX), the cohesive crossover (COHX), and finally the multiple parent crossover (MPX).

After the analysis of the experimental results, we obtained the crossovers ranking which shows relatively high performance of the crossovers with a lower degree of disruption, as well as the crossovers that incorporate the problem-oriented knowledge, including the multiple parent crossover. On the whole, three recombination operators (MPX, SPX, COHX) seem to be superior to the remaining operators. In particular, the swap path operator enabled discovering new best know solutions for eight GPP instances and solving all the GPP instances pseudo-optimally at surprisingly small computational times. Some more experiments would be useful in order to acknowledge that these operators are really the best. Nevertheless, MPX, SPX, and COHX may be recommended as promising recombination operators for the designers of new genetic algorithms for the GPP, QAP and similar combinatorial problems. The designers, however, must be careful by implementing the particular versions of the crossovers. They should not forget about exploiting the specific structure of a problem – it is of extreme importance by seeking near-optimal solutions.

The additional experimental results for the GPP demonstrate that EHGA is preferable to standard hybrid GAs where instruments of escaping stagnation of the search are missing or insufficient. These results support the opinion that is very helpful to use proper mechanisms for premature convergence avoidance in the presence of rapid improvement of the offspring.

There is still a room for the further modifications and enhancements of the crossover operators. This is especially true for the multiple parent crossover, which allows producing the offspring in many different ways. The investigation of new multiple parent operators could be one of the possible directions of the future research.

Acknowledgments

This work is supported by Lithuanian State Science and Studies Foundation (Grant No. T-06276).

The author thanks anonymous referee for the comments and suggestions that contributed to improve the quality of the paper.

References

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

Boese, K.D., A.B. Kahng and S. Muddu (1994). A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, **16**, 101–113.

Burkard, R.E., E. Çela, P.M. Pardalos and L. Pitsoulis (1998). The quadratic assignment problem. In D.Z. Du and P.M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, vol. 3. Kluwer, Dordrecht. pp. 241–337.

Çela, E. (1998). *The Quadratic Assignment Problem*: *Theory and Algorithms*. Kluwer, Dordrecht.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand, New York.

Drezner, Z. (2003). A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, **15**, 320–330.

Drezner, Z. (2005). Finding a cluster of points and the grey pattern quadratic assignment problem. *Working Paper*. College of Business and Economics, California State University, Fullerton, CA.

Fleurent, C., and J.A. Ferland (1994). Genetic hybrids for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz (Eds.), *Quadratic Assignment and Related Problems*. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16. AMS, Providence. pp. 173–188.

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

Glover, F. (1994). Genetic algorithms and scatter search: unsuspected potential. *Statistics and Computing*, **4**, 131–140.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading.

Goldberg, D.E., and R. Lingle (1985). 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. pp. 154–159.

Koopmans, T., and M. Beckmann (1957). Assignment problems and the location of economic activities. *Econometrica*, **25**, 53–76.

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

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

Migkikh, V.V., A.A. Topchy, V.M. Kureichik and A.Y. Tetelbaum (1996). Combined genetic and local search algorithm for the quadratic assignment problem. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications* (*EVCA'96*). Presidium of the Russian Academy of Sciences, Moscow. pp. 335–341.

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

Misevicius, A. (2003b). 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. pp. 598–609.

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

Misevicius, A. (2004b). An improved hybrid genetic algorithm: new results for the quadratic assignment problem. *Knowledge-Based Systems*, **17**, 65–73.

Misevicius, A. (2005). A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, **30**, 95–111.

Misevicius, A., and D. Rubliauskas (2005). Performance of hybrid genetic algorithm for the grey pattern problem. *Information Technology and Control*, **34**, 15–24.

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

Mühlenbein, H. (1989). Parallel genetic algorithm, population dynamics and combinatorial optimization. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo. pp. 416–421.

Oliver, I.M., D.J. Smith and J.R.C. Holland (1987). 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. pp. 224–230.

Pardalos, P.M., and M.G.C. Resende (Eds.) (2002). *Handbook of Applied Optimization*. Oxford University Press, New York.

Pardalos, P.M., and H. Wolkowicz (Eds.) (1994). *Quadratic Assignment and Related Problems*. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16. AMS, Providence.

Pitsoulis, L., and P.M. Pardalos (2001). Quadratic assignment problem, QAP. In C.A. Floudas and P.M. Pardalos (Eds.), *Encyclopedia of Optimization*, vol. 4. Kluwer, Dordrecht. pp. 405–436.

Radcliffe, N., and P. Surry (1994). Fitness variance of formae and performance prediction. In L.D. Whitley and M. Vose (Eds.), *Proceedings of the Third Workshop on Foundations of Genetic Algorithms*. Morgan Kaufmann, San Francisco. pp. 51–72.

Reeves, C.R., and J.E. Rowe (2001). *Genetic Algorithms*: *Principles and Perspectives*. Kluwer, Norwell.

Rendl, F. (2002). The quadratic assignment problem. In Z. Drezner and H. Hamacher (Eds.), *Facility Location*: *Applications and Theory*. Springer, Berlin. pp. 439–457.

Stützle, T. (1997). MAX-MIN ant system for quadratic assignment problems. *Research Report AIDA-97-04*, Darmstadt University of Technology, Darmstadt, Germany.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo. pp. 2–9.

Taillard, E. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science*, **3**, 87–105.

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

Tate, D.M., and A.E. Smith (1995). A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, **1**, 73–83.

A. Misevičius was born in 1962, in Marijampolė, Lithuania. He received dipl. eng. degree from Kaunas Polytechnic Institute, Lithuania, in 1986. A. Misevičius got Doctor degree in 1996, Kaunas Univ. Technol. He was conferred 3rd award in Young Scientists' Competition, Kaunas Univ. Technol, in 1997. A. Misevičius is currently assoc. prof. at Dept. of Practical Informatics, Kaunas Univ. Technol. Author and co-author of over 60 res. papers and acad. texts on different topics of computer science. The main research interests include: computer-aided design, design and applications of heuristics and meta-heuristics for combinatorial problems.

Eksperimentai su hibridiniu genetiniu algoritmu „pilkų šablonų (freimų)“ sudarymo uždaviniui

Alfonsas MISEVIČIUS

Pastaruoju metu genetiniai algoritmai (GA) yra tapę labai populiarūs sprendžiant įvairius kombinatorinio optimizavimo uždavinius. Šiame straipsnyje nagrinėjami hibridinio genetinio algoritmo efektyvumo klausimai vadinamajam „pilkų šablonų (freimų)“ sudarymo uždaviniui, kuris yra atskiras gerai žinomo kvadratinio paskirstymo uždavinio atvejis.

Pagrindinis dėmesys skirtas kryžminimo („krosoverio“) operatoriams. Pastarieji, kaip žinoma, yra gana svarbūs sudarant efektyvius genetinius algoritmus, galinčius sėkmingai „konkuruoti“ su kitais moderniais euristiniais algoritmais. Straipsnyje aptariama krosoverio operatorių bendri funkcionavimo principai, specifinės savybės, ypatumai. Kryžminimo operatoriai lyginami tiek konceptualiniu požiūriu, tiek eksperimentiškai. Eksperimentai atlikti su šiais kryžminimo operatoriais: „vieno taško“ krosoveriu, tolygiojo paskirstymo krosoveriu, dalinio atvaizdavimo krosoveriu, ciklo krosoveriu, sukeitimų krosoveriu, suliejimo krosoveriu. Išbandytas ir gana originalus kryžminimo būdas – daugelį sprendinių-tėvų naudojantis operatorius („kelių tėvų“ krosoveris).

Pateikiami atliktų tyrimų rezultatai, gauti eksperimentiškai palyginus minėtus kryžminimo operatorius „pilkų šablonų“ sudarymo uždaviniui. Viso spręsta 126 testiniai pavyzdžiai („gairės“). Rezultatai liudija, kad santykinai geresnės kokybės sprendiniai pasiekiami, panaudojant daugelio tėvų, sukeitimų, suliejimo operatorius. Dėl geresnių paieškos laiko charakteristikų paildomiems eksperimentams panaudotas sukeitimų krosoveris. Šių eksperimentų metu buvo surasti aštuoni sprendiniai, kuriems tikslo funkcijos reikšmės pasirodė esančios rekordinės – geresnės negu iki tol buvusios geriausios žinomos. Visiems 126 testiniams pavyzdžiams buvo gauti pseudo-optimalūs sprendiniai per labai trumpą hibridinio genetinio algoritmo vykdymo laiką.

Rekomenduojama sukeitimų, „kelių tėvų“ krosoverius, atitinkamai adaptavus, išbandyti ir kitiems kombinatorinio optimizavimo uždaviniams spręsti.