ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol. 34, No. 2

COMPARISON OF CROSSOVER OPERATORS FOR THE QUADRATIC ASSIGNMENT PROBLEM

Alfonsas Misevičius[1], Bronislovas Kilda[2]

Department of Practical Informatics[1], Computer Department[2], Kaunas University of Technology
Studentų St. 50–400a/416a[1], Studentų St. 50–412[2], LT–51368 Kaunas, Lithuania

Abstract. Crossover (i.e. solution recombination) operators play very important role by constructing competitive genetic algorithms (GAs). In this paper, the basic conceptual features and specific characteristics of various crossover operators in the context of the quadratic assignment problem (QAP) are discussed. The results of experimental comparison of more than 10 different crossover operators for the QAP are presented. The results obtained demonstrate high efficiency of the crossovers with relatively low degree of disruption, namely, the swap path crossover (SPX), the cohesive crossover (COHX), the one point crossover (OPX). Another promising operator is so-called multiple parent crossover (MPX) operator based on special type of recombination of several solutions-parents. The results from the experiments show that MPX operator enables to achieve better solutions than other operators tested.

Keywords: heuristic algorithms, genetic algorithms, crossover operators, quadratic assignment problem.

1. Indroduction

The quadratic assignment problem (QAP) can be formulated as follows. 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 $\{1, 2, ..., n\}$ be given. The goal is to find a permutation $\pi = (\pi(1), \pi(2), ..., \pi(n)) \in \Pi$ that minimizes

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

One of the interpretations of the QAP is the facility layout problem [14]. In this case, n is the number of facilities/locations, the element a_{ij} denotes the flow of materials from facility i to facility j, and b_{kl} can be seen as the distance between location k and location l. The permutation $\pi = (\pi(1), \pi(2), ..., \pi(n))$ then represents an assignment of n facilities to n locations (here, $\pi(i)$ ($\pi(i) \in \{1, 2, ..., n\}$) is the location what facility i is assigned to).

It has been proved that the QAP is NP-hard, therefore various heuristic approaches are used for solving medium- and large-scale QAPs within reasonable computation times. For surveys of the heuristics for the QAP, see [3,5]. Starting from 1994, genetic algorithms (GAs), their modifications and hybrids are among the advanced heuristic techniques for this problem (see, for example, [1,8,15,18,21]).

Very roughly, genetic algorithms can be characterized as follows [12]. Let P be a subset of Π; it is referred to as a population, and it is composed of individuals, i.e. solutions (permutations in the context of the QAP), $\pi_1, \pi_2, ... \pi_{PS=|P|}$. Each individual ($\pi_i$) is associated with a fitness, i.e. the corresponding objective function value ($z(\pi_i)$). 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 solutions of P is selected to be parents by use of a selection mechanism. 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 undergo a mutation (random perturbation) to prevent a premature loss of the diversity within the population. Over many steps, i.e. 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. The best-survived individual is regarded as a result of the genetic algorithm.

For a more complete discussion on the principles of GAs, the reader is addressed to [6,12,27].

It should be noted that the state-of-the-art genetic algorithms are rather hybrid (combined genetic local search) algorithms which incorporate additional heuristic components [24]. Typically, a post-crossover

(or post-merging) procedure is used to play the role of a local improvement algorithm applied to the solution previously produced by the crossover. This way of proceeding ensures that the population consists solely of locally optimal solutions. However, despite the optimized populations, the recombination of solutions still remains one of the critical things by constructing competitive genetic algorithms. Very likely, the role of recombination operators within hybrid genetic algorithms (HGA) is more significant than in the ordinary GAs. In fact, we can think of HGA as a process that combines intensification and diversification (I&D) of the search (for more details on I&D methodology, see, for example, [22]). The intensification (local improvement algorithm) concentrates the search in certain local (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. From this point of view, the crossover is a special sort of the diversification (solution reconstruction) mechanism, which – generally speaking – guides the global search, i.e. exploration of new and new regions of the solution space. The proper exploration mechanism is, in some sense, even more severe than intensification, and may add crucial influence on the resulting efficiency of the search.

In this paper, we discuss issues related, namely, to the investigation of crossover operators within HGAs in the context of the quadratic assignment problem. Both conceptual and experimental comparison of various types of crossovers for the QAP are given. More precisely, in Section 2, the basic features of twelve different crossover operators for the QAP are overviewed. Then, the computational results for these crossovers are presented in Section 3. Section 4 completes the paper with concluding remarks.

2. Discussion of crossover operators for the QAP

As mentioned above, 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 parents' characteristics and creation of entirely new features. Mathematically, crossover can be defined as a binary operator (function) ψ: $\Pi \times \Pi \to \Pi$ 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, $\pi'(i) = \pi''(i) \Rightarrow \pi^{o}(i) = \pi'(i) = \pi''(i)$, $i = 1, 2, ..., n$, where π', π'', π^{o} are the parents and offspring, respectively. The inheritance of the remaining genes can be accomplished in a variety of ways. These ways, with respect to the quadratic assignment problem, are discussed below.

We start our overview of the crossover operators with the crossover by Tate and Smith proposed as far back as 1995 [29]. This crossover can be viewed as some kind of uniform crossover adapted to permutation-based solutions. So, we call it as uniform like crossover (ULX). It 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 Figure 1).

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

Figure 1. Example of uniform like crossover

Uniform crossover allows some flexibility, and different variations of the basic procedure are possible. Below, we give three modifications/enhancements of the basic ULX operator. In some sense, these modifications can also be considered as special kind crossovers. We will call them: randomized ULX (RULX), modified ULX (or block crossover (BX)), and extended ULX (or repair crossover (RX)).

The only difference between the standard uniform crossover described above and the randomized one is in the manner how the positions of solutions are scanned. In standard ULX, the order of scanning is fixed – from left to right. In RULX, the consideration of the positions (both the assigned and unassigned) is done in a random way. The motivation of this technique is adding more randomness and diversity to the offspring generation process (this is important by avoiding a premature convergence).

Another modification of ULX – we call it block crossover – is distinguishing for the fact that some blocks (segments) of elements are considered, instead of the single elements. The block size BS is in the range $[1, \lfloor n/2 \rfloor]$. Note that blocks may be of different sizes; for example, if $n = 9$ and the number of blocks is equal to 4, then one obtains three blocks of size 2 and one block of size 3. By copying blocks, the feasibility of permutation must be kept (see Figure 2). $BS = \lfloor n/2 \rfloor$ means that the first segment of size $\lfloor n/2 \rfloor$ is

copied from one of the parents (say, π') to the offspring, the remaining items are copied from the "opposite" parent (π'')) in such a way that the feasibility of the resulting solution is preserved. If there still exist unassigned positions in the offspring, then the missing elements can be taken from either the first or second parent.

The central idea of the extended crossover is the combination of ULX and a local improvement-based repair (correction) procedure applied to the offspring produced by ULX. More precisely, one tries to improve the offspring by pairwise interchanges of the elements – but only those that are not inherited from the parents. A candidate list CL is created, where $CL(i) = \pi^o(i)$, $CL(i) \neq \pi'(i)$, $CL(i) \neq \pi''(i)$ (π', π'', π^o are the parents and offspring, respectively). The members of the candidate list take part in the improvement process (Of course, if the candidate list size is zero ($|CL| = 0$), the improvement process is omitted.) The template of the corresponding procedure is given in Figure 3.

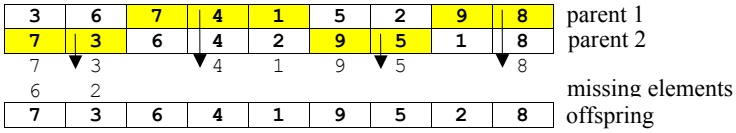

Figure 2. Example of block crossover

procedure *partial_steepest_descent*

// Let π be the current solution-offspring. At each step of procedure, an attempt is made to
// replace the current solution by the solution that improves most the objective function value.
// The process is continued until no improving solution exists

repeat

$$\pi^\bullet := \arg\min_{\substack{i=CL(1),\ldots,CL(w-1)\\ j=i+1,\ldots,CL(w)}} z(\pi \oplus p_{ij});$$

if $z(\pi^\bullet) < z(\pi)$ **then** $\pi := \pi^\bullet$ // replace the current solution by the new one

until $\pi^\bullet \neq \pi$

end // *partial_steepest_descent*

Figure 3. Template of partial local improvement procedure used in repair crossover.
Notes. 1. p_{ij} is the elementary perturbation operator which simply exchanges ith and jth elements in the current permutation; in this case, the expression $\pi \oplus p_{ij}$ denotes the permutation that is obtained from the current permutation π by applying p_{ij}. 2. $w = |CL|$

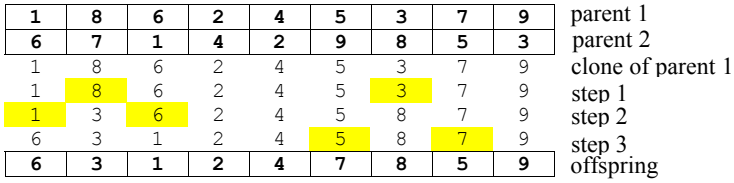

Figure 4. Example of uniform partially mapped crossover

The experiments have shown that the size of the candidate list is much more less than n in most cases; so, the partial local improvement procedure does not contribute considerably to the total amount of run time of the genetic algorithm.

Further, let us describe a variant of the well-known recombination operator, the partially-mapped crossover (PMX). The key idea of PMX is that it works with a part of a chromosome – mapping section – located between two crossover points (sites) [13]. PMX has been proven to be highly effective for the traveling salesman problem [13], however the straightforward PMX procedure does not work well for the QAP. For this reason, Migkikh et al. [19] proposed a modification of PMX based on using a number of random mapping points – instead of one mapping segment. This crossover was referred to as a uniform PMX (UMPX). The basic steps of UPMX are as follows: a) clone the offspring π^o 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. $\pi^o(pos_2) = \pi''(pos_1)$; d) swap the content of $\pi^o(pos_1)$ and $\pi^o(pos_2)$; e) repeat steps a-e k times, where $k = \lfloor n/3 \rfloor$ (according to [19]). An illustration of UPMX is presented in Figure 4.

In [16], Merz and Freisleben introduced a distance preserving crossover (DPX) for the QAP. (The earlier version of this crossover was tried on the traveling

salesman problem [10].) The main idea of DPX is that this recombination operator aims at producing the offspring which has the same distance to each of its parents, and this distance is equal to the distance between the parents themselves. Remind that the "distance" ρ between two permutations π_1 and π_2 is defined as a number of the elements that are assigned to different positions in the corresponding permutations, i.e. $\rho(\pi_1,\pi_2) = \left|\{i \mid \pi_1(i) \neq \pi_2(i)\}\right|$. So, the elements that are identical for the same positions in both parents (π' and π'') will be copied to the offspring (π°). The contents for all other genes change; that is, the remaining positions are randomly filled in with the yet unassigned elements, taking care that no assignment that appears in just one parent is copied to the child. A simple example of how DPX works is shown in Figure 5.

We continue consideration of the crossover operators by concerning the principle of a cycle crossover (CX) [17,18,25]. The key idea of this operator 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 (mutation) occurs. Then, the next unassigned location is processed in the same manner until all the locations have been considered. An example is presented in Figure 6.

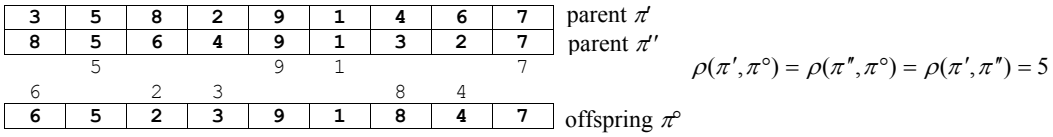

Figure 5. Example of distance preserving crossover

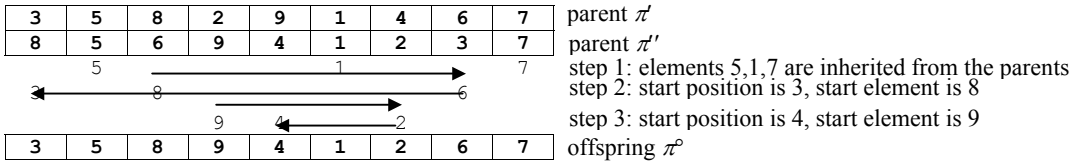

Figure 6. Example of cycle crossover

Ahuja et al. used a swap path crossover (SPX) in their "greedy genetic algorithm" [1]. (Note that the original idea of this type of crossover was developed by Glover [11], who referred to it as a "path relinking".) 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 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. propose 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 swap path crossover is illustrated in Figure 7. For more details, see [1].

One point crossover (OPX) operators [12] are classical solution recombination procedures widely used in early versions of genetic algorithms. One of the variants of OPX for the QAP is due to Lim et al. [15]. The idea of OPX is quite simple. A crossing point (site) is chosen randomly between 1 and $n - 1$ in one of the parents. As a result, a child chromosome is obtained, containing information partially determined by each of parent chromosomes. Again, the care of preserving of the solution feasibility should be taken (see Figure 8).

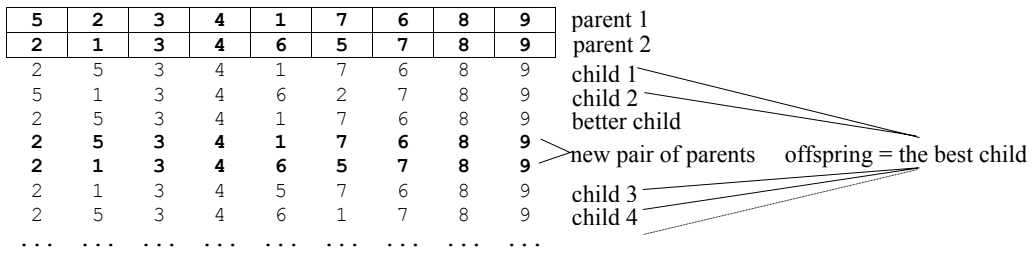

Figure 7. Example of swap path crossover

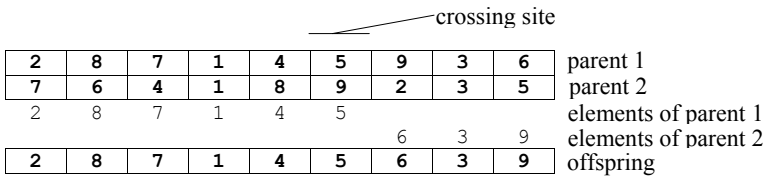

Figure 8. Example of one point crossover

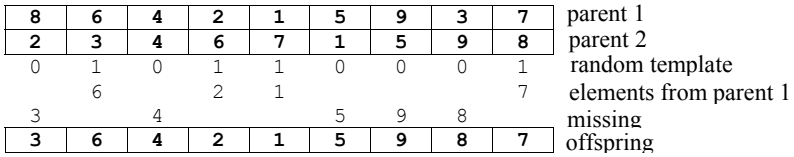

Figure 9. Example of order-based crossover

So-called order-based operators [7] have been applied successfully for problems as scheduling, where the objective is to obtain the order (sequence) in which jobs are assigned by the schedule. The basic feature of order-based crossovers (OBX) is that they preserve the relative order of the alleles in chromosomes. So, a number of items (elements) are selected from one of the parents and copied to the offspring. The missing elements (alleles) are taken from the other parent in order. An example of this crossover is shown in Figure 9 (see also [30] for more details).

Recently, Drezner introduced a quite original solution recombination operator – a cohesive crossover (COHX) [8]. COHX produces the offspring in several steps. At the beginning, some mask – an $n_1 \times n_2$ matrix $\boldsymbol{M}$ – is created. n_1 and n_2 are chosen in such a way that $n_1 \cdot n_2 = n \wedge n_1 + n_2 \to \min$. The initial mask position is fixed at (i_0, j_0), where $i_0 \in \{1, 2, ..., n_1\}$, $j_0 \in \{1, 2, ..., n_2\}$. The mask matrix is then filled in according to wave propagation fashion (see Figure 10). There exist n different masks $\boldsymbol{M}^{(1)}$, $\boldsymbol{M}^{(2)}$, ..., $\boldsymbol{M}^{(k)}$, ..., $\boldsymbol{M}^{(n)}$, where k, i_0, and j_0 are in the following relation: $k = n_2 \cdot (i_0 - 1) + j_0$, $i_0 = 1, 2, ..., n_1$, $j_0 = 1, 2, ..., n_2$. Then, the kth recombined solution $\pi^{(k)}$ ($k \in \{1, 2, ..., n\}$) is generated in three steps:

1) $$\pi^{(k)}(i) = \begin{cases} \pi_{better}(i), \boldsymbol{M}^{(k)}(((k-1) \operatorname{div} n_2) + 1, \\ \qquad ((k-1) \bmod n_2) + 1)) \le \eta \\ 0, \text{otherwise} \end{cases};$$

where $i = 1, 2, ..., n$, $\pi_{\text{better}} = \operatorname{argmin}\{z(\pi'), z(\pi'')\}$, π', π'' are the solutions-parents, and η is the median of $\boldsymbol{M}^{(k)}$, i.e. $\eta = \frac{\sum_{i}^{n_1}\sum_{j}^{n_2} \boldsymbol{M}^{(k)}(i,j)}{n_1 \cdot n_2}$;

2) $$\pi^{(k)}(i) = \begin{cases} \pi^{(k)}(i), \pi^{(k)}(i) > 0 \\ \pi_{worse}(i), \pi^{k}(i) = 0 \wedge \pi_{worse}(i) \text{ not in } \pi^{(k)} \\ 0, \text{otherwise} \end{cases};$$

where $i = 1, 2, ..., n$, $\pi_{\text{worse}} = \operatorname{argmax}\{z(\pi'), z(\pi'')\}$;

3) for every unassigned position i ($\pi^{(k)}(i) = 0$), an item is chosen randomly from those not yet included in the offspring.

A visual example of generation of a solution is given in Figure 11. As a result, n solutions are produced, but only the best of them is regarded as an offspring, i.e. $\pi^{\circ} = \underset{k=1,2,...n}{\arg\min}\, z(\pi^{(k)})$.

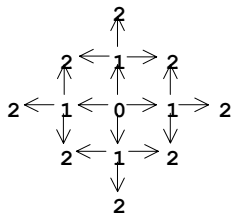

Figure 10. Filling in a mask

Multiple parent crossover (MPX) was described by Misevičius in [23], although the idea of using combinations of several solutions goes back to Boese et al. [2] and Fleurent and Glover [9]. MPX is distinguished for the fact that the offspring derives the information from many parents – this is the contrast and, at that time, advantage to the traditional operators, where useful information may be left out of account because of using two parents only. In MPX, the ith element, i.e. allele of the offspring π° is created by choosing such a number j (among those not yet chosen) that probability that $\pi^{\circ}(i) = j$ $\Pr(\pi^{\circ}(i) = j)$ is maximized. Here, the probability $\Pr(\pi^{\circ}(i) = j)$ is equal to $\frac{d_{ij}}{\sum_{j}^{n} d_{ij}}$, where d_{ij} is the entry of a desirability matrix $\boldsymbol{D} = (d_{ij})_{n \times n}$, and is equal to 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). The process is to be continued until all the genes of the offspring take on their values. An example of producing of the offspring in multiple parent crossover ($\mu = 5$) is given in Figure 12.

Note that, in this work, the desirability matrix was slightly modified. Instead of $\boldsymbol{D} = (d_{ij})_{n \times n}$, we used $\boldsymbol{D}' = (d'_{ij})_{n \times n}$, where $d'_{ij} = d_{ij} + \varepsilon$, here ε is a correction (noise).

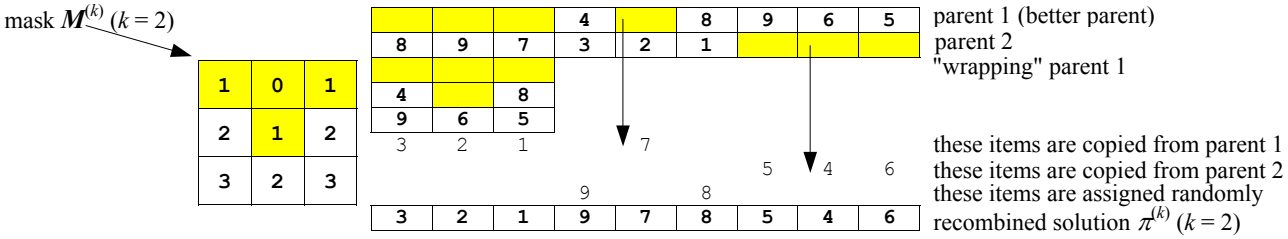

Figure 11. Example of cohesive crossover

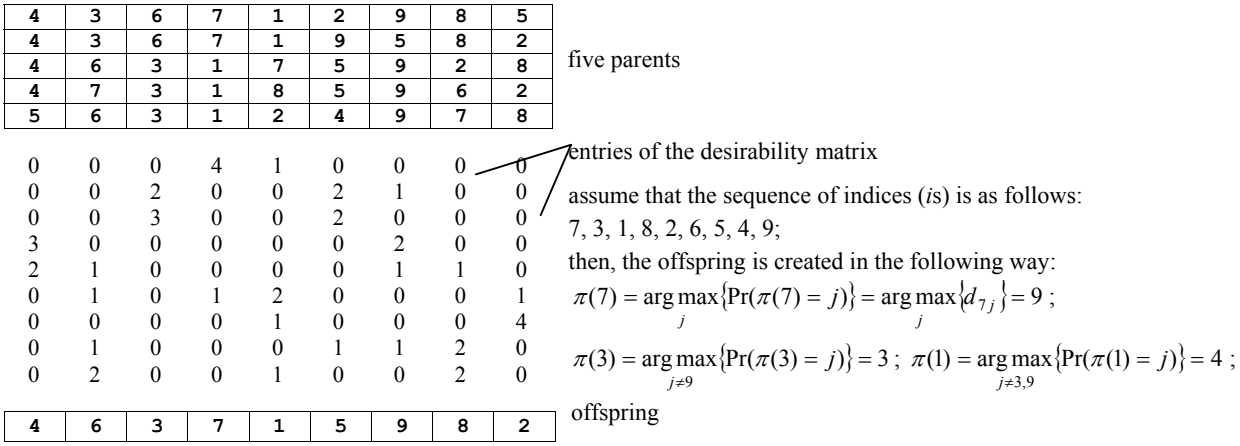

Figure 12. Example of multiple parent crossover

The recombination operators discussed can be categorized depending on various criteria (factors): the number of parents, the level (degree) of randomness (distortion), the problem oriented knowledge, time/memory complexity, implementation/programming aspects, etc. Let us concern the randomization factor in more details. Very roughly speaking, this factor can be viewed as a measure of how "far" is the offspring from the parents. Regarding this factor, the crossover operators can be classified as less disruptive and more disruptive. Radcliffe and Surry [26] use the terms "assorting" and "respectful", respectively. (Note that most of the above crossovers may be viewed as "respectful"; the only cycle crossover falls into the category of "assorting" operators.) In general, there are two situations in merging process: explicit mutation and implicit mutation. The "assorting" crossover is characterized by explicit mutation, while implicit mutation is an indication of the "respectful" operator. As an explicit mutation, we call the situation where the offspring is different from both first and second parent, however every allele of the child is from the corresponding gene of either first or second parent; that is, $\pi^{\circ} \neq \pi' \wedge \pi^{\circ} \neq \pi'' \wedge (\pi^{\circ}(i) = \pi'(i) \vee \pi^{\circ}(i) = \pi''(i)$, $i = 1, 2, ..., n)$. This is a very strict condition. It is the reason of why the assorting recombination is hardly accomplished for some problems. Meanwhile, implicit mutation offers more "degree of freedom"; the only requirement to fulfil is that the offspring is different from the parents (provided that the offspring necessary 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 corresponding element (allele) $(\pi^{\circ}(i))$ can be considered as "foreign" element, since this element is not contained in (one of) the parents. The number of foreign elements may be seen as an indicator of "disruptiveness" of the crossover: the more number of foreign elements, the more disruptive crossover operator.

It can be guessed that implicit mutations have a positive effect on the performance of the genetic search, especially, in the cases where highly elaborated post-crossover algorithms are used. The results obtained (see Section 3) confirmed this preliminary conjecture: crossovers with implicit mutation outperformed the representative of explicit operators, the cycle crossover, for most of the QAP instances tested. However, the crossover designer must be very careful by preferring a "respectful" crossover to an "assorting" one: for the particular problems, implicit mutations may appear to be inferior to explicit ones.

The other factors of crossovers seem to be more intuitively clear. These factors (together with the randomization level characteristic) are summarized in Table 1.

Table 1. Crossover characteristics: towards conceptual comparison of crossovers

Crossovers / Characteristics	ULX,RULX, BX,OBX	RX	UPMX	DPX	CX	SPX	OPX	COHX	MPX
Number of parents	2	2	2	2	2	2	2	2	≥2
Randomization level (explicit mutation (EM), implicit mutation (IM))	IM	IM	IM	IM	EM	IM	IM	IM	IM
Time complexity	$O(n)$	$O(n+w^2)$	$O(n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Memory size complexity	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$
Implementation (programming) aspects (TI – rather trivial implementation, NTI – rather non-trivial implementation)	TI	NTI	TI	NTI	NTI	TI	TI	NTI	NTI
Problem oriented knowledge (embedded (E), not embedded (NE))	NE	E	NE	NE	NE	E	NE	E	E
Additional properties			$\rho(\pi',\pi^\circ) \le 2k$, where k is the number of swaps	high degree of disruption of the offspring		low degree of disruption	relatively low degree of disruption; $\rho(\pi',\pi^\circ) \le n-c$, where c is the crossing position	relatively low degree of disruption	

3. Computational experiments

In this section, we present the results of experimental comparison of the crossovers outlined above. In the experiments, we used the instances (benchmarks) of the QAP taken from the well-known library QAPLIB [4]. The types of the instances we examined are as follows:

(a) random instances (these instances are randomly generated according to a uniform distribution; in QAPLIB, they are denoted by tai20a, tai25a, tai30a, tai35a, tai40a, tai50a, tai60a, tai80a, and tai100a; the corresponding numeral (20, 25, and so on) in the instance name denotes the size of the instance);

(b) real-life like instances (instances of this type are generated in such a way that the entries of the matrices ***A*** and ***B*** resemble the distribution from real world problems; these instances are denoted by tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, tai60b, tai80b, tai100b, and tai150b).

As an experimental basis for the crossover operators, we used an extended hybrid genetic algorithm (EHGA). The basic flowchart of EHGA is shown in Figure 13. The details of this algorithm are presented in Misevičius' paper [20]. Remind that the high performance of this algorithm was achieved by effective implementation of the genetic-tabu search paradigm, i.e. combining the genetic operators with the iterated tabu search (ITS) procedure. The details of the ITS procedure can be found in [22].

The efficiency measure for crossover operators is the average deviation of solutions obtained from the best known solution – $\bar{\delta}$ ($\bar{\delta} = 100(\bar{z} - \breve{z})/\breve{z}\,[\%]$, where $\bar{z}$ is the average objective function value over 10 restarts (single applications of EHGA to a given instance), and $\breve{z}$ is the best known value (BKV) of the objective function).

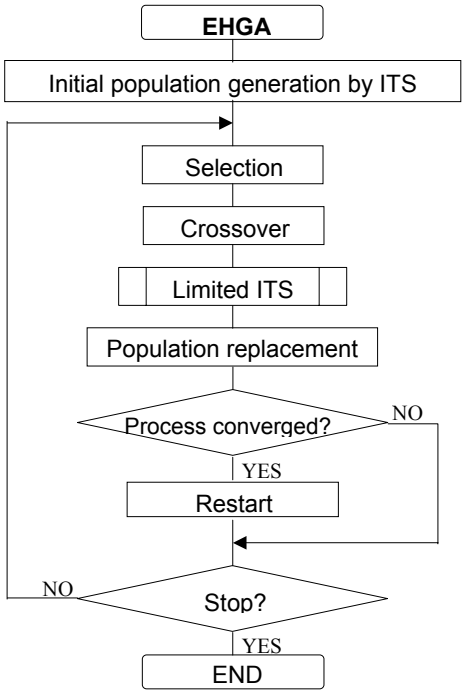

Figure 13. Generalized flowchart of the extended hybrid genetic algorithm

In the experimental comparison, equated conditions are created: all the crossover variants use the identical initial solutions and require approximately the same CPU time. For the sake of more fairness, we carried out two sets of experiments (shorter runs and longer runs, with smaller and larger number of generations, respectively). The following are the values of the control parameters of EHGA (of course, they are equivalent for all the crossovers compared): population size – $\lfloor \sqrt{n} \rfloor$; number of generations – $n/3$ (for the first set of experiments) and $3n$ (for the second set); number of offsprings (crossovers) per generation – 1; number of iterations of the post-crossover, i.e. the iterated tabu search (ITS) procedure – $\frac{1}{10}n^2$ (for the random instances) and $4n$ (for the real-life like instances). The number of parents in the MPX crossover is equal to the population size.

The results of comparison of the crossovers are presented in Tables 2–5.

Table 2. Comparison of the crossover operators for the QAP: shorter run results for the random instances. The best results obtained are printed in bold face. CPU times per restart are given in seconds. 900 MHz PENTIUM computer was used in the experiments

Instance	BKV	$\bar{\delta}$										CPU time
		BX	RX	UPMX	DPX	CX	SPX	OPX	OBX	COHX	MPX	
tai20a	703482 [a]	0.730	0.666	0.680	0.718	0.652	0.622	0.622	0.681	0.655	**0.615**	*0.1*
tai25a	1167256 [a]	1.005	0.710	0.754	1.090	0.923	0.899	0.717	0.882	0.899	**0.698**	*0.3*
tai30a	1818146 [a]	0.685	0.648	0.525	0.827	0.603	0.668	0.531	0.693	**0.496**	0.636	*0.6*
tai35a	2422002 [a]	0.754	0.734	**0.659**	0.779	0.695	0.814	0.763	0.711	0.698	0.687	*1.2*
tai40a	3139370 [a]	0.727	0.810	0.838	0.863	0.786	0.765	0.752	0.770	0.766	**0.708**	*2.7*
tai50a	4941410 [a]	0.964	0.918	0.961	0.966	0.901	**0.865**	0.945	0.919	0.889	0.904	*8.8*
tai60a	7205962 [b]	0.833	0.866	0.876	0.878	0.833	0.828	0.850	0.841	0.848	**0.819**	*21*
tai80a	13546960 [b]	0.499	0.498	0.506	0.522	0.458	**0.435**	0.458	0.518	0.505	0.455	*85*
tai100a	21123042 [b]	0.377	0.411	0.402	0.406	0.370	**0.314**	0.377	0.403	0.385	0.329	*250*
Average:		0.730	0.696	0.689	0.783	0.691	0.690	0.668	0.713	0.682	**0.650**	

[a] comes from [4]; [b] comes from [22].

Table 3. Comparison of the crossover operators for the QAP: shorter run results for the real-life like instances. The best results obtained are printed in bold face. CPU times per restart are given in seconds.900 MHz PENTIUM computer was used in the experiments

Instance	BKV	$\bar{\delta}$										CPU time
		RULX	RX	UPMX	DPX	CX	SPX	OPX	OBX	COHX	MPX	
tai20b	122455319 [a]	0.090	**0.045**	0.090	0.101	0.085	0.090	0.090	0.091	**0.045**	0.090	*0.1*
tai25b	344355646 [a]	0.150	0.201	0.194	0.206	0.166	**0.045**	0.273	0.079	0.175	0.115	*0.2*
tai30b	637117113 [a]	**0.315**	0.410	0.549	0.690	0.514	0.551	0.371	0.560	0.528	0.353	*0.3*
tai35b	283315445 [a]	0.289	0.287	**0.163**	0.368	0.240	0.281	0.271	0.218	0.242	0.289	*0.4*
tai40b	637250948 [a]	0.550	0.420	0.444	0.411	0.464	**0.068**	0.352	0.515	0.414	0.243	*0.9*
tai50b	458821517 [a]	0.285	0.261	0.228	0.500	0.205	0.168	0.296	0.223	**0.121**	0.316	*2.4*
tai60b	608215054 [a]	0.184	0.181	0.176	0.330	0.183	**0.163**	0.176	0.201	0.166	0.173	*4.3*
tai80b	818415043 [a]	**0.337**	0.477	0.594	0.909	0.547	0.448	0.381	0.474	0.396	0.338	*15*
tai100b	1185996137 [a]	0.357	0.422	0.387	0.644	0.458	0.471	0.429	0.370	0.313	**0.290**	*35*
tai150b	498896643 [b]	0.639	0.622	0.684	1.020	0.698	0.674	0.592	0.599	0.623	**0.557**	*120*
Average:		0.320	0.333	0.351	0.518	0.356	0.296	0.323	0.333	0.302	**0.276**	

[a] comes from [4]; [b] comes from [28].

Table 4. Comparison of the crossover operators for the QAP: longer run results for the random instances. The best results obtained are printed in bold face. CPU times per restart are given in seconds.900 MHz PENTIUM computer was used in the experiments

Instance	BKV	$\bar{\delta}$										CPU time
		BX	RX	UPMX	DPX	CX	SPX	OPX	OBX	COHX	MPX	
tai20a	703482	0.216	0.216	0.238	0.221	0.268	0.207	**0.202**	0.268	0.229	0.246	*1.0*
tai25a	1167256	0.190	0.191	**0.117**	0.231	0.158	0.151	0.190	0.159	0.169	0.150	*3.0*
tai30a	1818146	0.091	0.097	0.111	0.074	0.040	0.091	0.120	0.112	0.091	**0.035**	*6.0*
tai35a	2422002	**0.175**	0.388	0.284	0.433	0.312	0.283	0.209	0.324	0.279	0.194	*12*
tai40a	3139370	0.454	0.474	0.455	0.503	0.424	0.444	0.441	0.425	0.417	**0.374**	*27*
tai50a	4941410	0.638	0.609	**0.565**	0.693	0.627	0.585	0.602	0.579	0.638	0.583	*88*
tai60a	7205962	0.608	0.629	0.623	0.689	0.619	0.609	0.596	0.613	0.651	**0.572**	*210*
tai80a	13546960	0.219	0.295	0.265	0.330	0.251	0.240	0.268	0.260	0.266	**0.218**	*850*
tai100a	21123042	0.192	0.203	0.191	0.230	0.165	0.142	0.178	0.219	0.195	**0.108**☼	*2500*
Average:		0.309	0.345	0.317	0.378	0.318	0.306	0.312	0.329	0.326	**0.276**	

☼ During the experimentation with MPX on the instance tai100a, we were successful in discovering new record-breaking solution. The new objective function value, which is better than that reported in [22], is equal to **21090402**.

Table 5. Comparison of the crossover operators for the QAP: longer run results for the real-life like instances. The best results obtained are printed in bold face. CPU times per restart are given in seconds.900 MHz PENTIUM computer was used in the experiments

Instance	BKV	$\bar{\delta}$										CPU time
		RULX	RX	UPMX	DPX	CX	SPX	OPX	OBX	COHX	MPX	
tai20b	122455319	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*1.0*
tai25b	344355646	**0.000**	**0.000**	**0.000**	0.007	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*2.0*
tai30b	637117113	**0.000**	0.018	0.001	0.019	0.017	0.001	0.001	**0.000**	**0.000**	0.001	*3.0*
tai35b	283315445	0.049	**0.014**	0.019	0.047	0.042	0.029	0.062	0.037	0.038	0.019	*4.0*
tai40b	637250948	**0.000**	**0.000**	**0.000**	0.045	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*9.0*
tai50b	458821517	0.035	0.003	0.006	0.088	0.002	0.012	**0.000**	**0.000**	**0.000**	0.014	*24*
tai60b	608215054	0.024	0.004	0.012	0.025	0.007	**0.002**	**0.002**	0.015	0.009	0.010	*43*
tai80b	818415043	**0.004**	0.067	0.119	0.173	0.013	0.017	0.045	0.017	0.015	0.016	*150*
tai100b	1185996137	0.104	0.091	0.117	0.145	0.108	0.091	0.113	0.114	0.079	**0.064**	*350*
tai150b	498896643	0.305	0.243	0.410	0.608	0.360	**0.232**	0.261	0.338	0.309	0.241	*1200*
Average:		0.052	0.044	0.068	0.116	0.055	0.038	0.048	0.052	0.045	**0.037**	

Table 6. Summarizing results of the experimental comparison of the crossover operators for the QAP

Instance groups	Deviation from BKV averaged over two sets of experiments										
	RULX	BX	RX	UPMX	DPX	CX	SPX	OPX	OBX	COHX	MPX
tai∗a		0.520	0.521	0.503	0.581	0.505	0.498	0.490	0.521	0.504	**0.463**
tai∗b	0.186		0.189	0.210	0.317	0.206	0.167	0.186	0.193	0.174	**0.157**
Average:			0.355	0.357	0.449	0.356	0.333	0.338	0.357	0.339	**0.310**

Some observations, looking at the results in Tables 2–5, are as follows. 1. Shorter run results are quite "flat", so it may be complicated to draw right conclusions; meanwhile, longer run results (obtained by large number of generations) are differentiated in a higher degree and allow to judge more about the efficiency of crossover operators. 2. Long time results demonstrate that crossovers have considerable influence

on the final solutions produced by the genetic algorithm, even in the cases when the powerful post-crossover procedures are used. This indicates that the recombination operators, which are responsible for the exploration of new regions in the solution space, hide high potential. 3. The performance of different crossover operators varies in large ranges; nevertheless some regularities can be discovered. For example, less disruptive crossovers appear to be more efficient than highly disruptive crossovers; surprisingly, the cycle crossover (the minimally available disruptive crossover) produces only medium-quality results. It can also be seen that the crossovers which incorporate the problem-oriented knowledge seem to be in average better than the remaining ("pure") operators. 4. Also, it can be observed that there exists some dissipation of results for the random and real-life like problems: typically, some of crossovers (except MPX and, maybe, SPX and OPX operators) work well on the random instances, but produce worse solutions for the real-life like instances, and vice versa. It is even more surprising that the MPX operator outperforms all the other competitors for both random and real-life like instances. This is a clear indication of high stability and universality of the multiple parent crossover. 5. The preliminary overall ranking of the crossover operators (sorted according to decreasing quality of solutions) for the random instances looks as follows: **MPX-OPX-SPX-UPMX**-COHX-CX-BX-RX-OBX-DPX. The similar ranking for the real-life like problems is as follows: **MPX-SPX-COHX-RULX-OPX**-RX-OBX-CX-UPMX-DPX. The resulting ranking is: **MPX-SPX-OPX-COHX**-RX-CX-UPMX-OBX-DPX (see Table 6). (It was somewhat unexpected that OPX produced slightly better results than COHX, which, in turn, was shown by Drezner [8] to be very effective for pseudo-random grid-based QAP instances. Thus, some more experiments would be useful in order to acknowledge the above ranking as really fair.) To summarize, SPX, OPX, COHX, and, especially, MPX appear to be superior to the remaining crossovers with respect to the QAP instances tested and could be recommended as perfect recombination operators for the designers of new (hybrid) genetic algorithms for the QAP.

4. Concluding remarks

In this paper, the solution recombination, i.e. crossover operators in the context of the quadratic assignment problems are discussed. These operators are known as playing an important role by developing robust genetic algorithms. The crossover procedures are considered in two aspects. Firstly, the basic principles (factors) and specific features of various recombination operators are outlined, and the table of crossover characteristics is given; then the results of the experimental comparison within the framework of hybrid genetic-tabu search algorithm are presented.

We implemented twelve different crossover procedures and their modifications in order to test the influence of the recombination operators to the genetic search process when applied to the quadratic assignment problem. The following crossover operators have been used in the experimentation: the uniform like crossover (ULX) and its modifications (the randomized ULX crossover (RULX), the ULX crossover combined with repair procedure (RX), the block crossover (BX)), the uniform partially-mapped crossover (UPMX), the distance preserving crossover (DPX), the cycle crossover (CX), the swap path crossover (SPX), the one point crossover (OPX), the order-based crossover (OBX), the cohesive crossover (COHX), and, finally, the multiple parent crossover (MPX). The results obtained from the experiments with the test instances of the QAP show high performance of the crossovers with lower degree of disruption and, especially, the multiple parent crossover. These robust crossovers incorporated into the hybrid genetic-tabu search algorithm resulted in encouraging results within small computation time. New best known solution for the QAP instance tai100a was obtained by using, namely, the MPX operator.

Further elaboration of the crossover operators mentioned with the focus on the multiple parent recombination, as well as development of innovative crossover operators for the QAP and similar combinatorial problems may be the subject of the future research.

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

[3] 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.

[4] R.E. Burkard, S. Karisch, F. Rendl. QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 1997, *Vol*.10, 391–403. [See also `http://www.seas.upenn.edu/qaplib/`.]

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

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

[7] L. Davis. Order-based genetic algorithms and the graph coloring problem. *In L.Davis (ed.), Handbook of Genetic Algorithms, Van Nostrand, New York*, 1991, 72–90.

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

[9] **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.

[10] **B. Freisleben, P. Merz.** A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *Proceedings of the* 1996 *IEEE International Conference on Evolutionary Computation, Nagoya, Japan*, 1996, 616–621.

[11] **F. Glover.** Genetic algorithms and scatter search: unsuspected potential. *Statistics and Computing*, 1994, *Vol*.4, 131–140.

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

[13] **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.

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

[15] **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.

[16] **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.

[17] **P. Merz, B. Freisleben.** A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. *In P.Angeline (ed.), Proceedings of* 1999 *Congress on Evolutionary Computation (CEC'*99*)*, *IEEE Press, New York*, 1999, 2063–2070.

[18] **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.

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

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

[21] **A. Misevicius.** An improved hybrid genetic algorithm: new results for the quadratic assignment problem. *Knowledge-Based Systems*, 2004, *Vol*.17, 65–73.

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

[23] **A. Misevicius, D. Rubliauskas.** Performance of hybrid genetic algorithm for the grey pattern problem. *Information Technology and Control*, 2005, *Vol*.34, *No*.1, 15–24.

[24] **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.

[25] **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.

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

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

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

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

[30] **M. Vázquez, L.D. Whitley.** A hybrid genetic algorithm for the quadratic assignment problem. *In L.D. Whitley, D.E. Goldberg, E. Cantú-Paz et al. (eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'*00*), Morgan Kaufmann, San Francisco*, 2000, 135–142.

DOI: 10.5755/j01.itc.34.2.11999