ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2008, Vol. 37, No. 3

ENHANCED IMPROVEMENT OF INDIVIDUALS IN GENETIC ALGORITHMS

Alfonsas Misevičius, Dalius Rubliauskas

Kaunas University of Technology, Department of Multimedia Engineering
Studentų St. 50, LT–51368 Kaunas, Lithuania

Abstract. In this paper, a new modification of the genetic algorithms (GAs) based on an enhanced improvement of individuals is discussed. The basic philosophy of the proposed approach is to accelerate the convergence speed of the genetic search by maintaining compact populations of the outstanding quality individuals – "super-individuals". The super-individuals are obtained by means of powerful iterated local search techniques. The increase in time for the improvement of individuals is compensated by decreasing the size of populations. We tested our approach on a well-known combinatorial optimization problem, the quadratic assignment problem (QAP). The results of the experiments show that using the enhanced improvement in GAs makes it possible to achieve very encouraging performance.

Keywords: combinatorial optimization, heuristics, genetic algorithms, quadratic assignment problem.

Introduction

Genetic algorithms (GAs) have been proven to be a powerful tool in various areas of computer science, including machine learning, search, and optimization. Although the principles of GAs were developed more than thirty years ago [13], the investigations of these algorithms still are an active area of research [12, 23, 25].

In this paper, we are concentrating on enhancing the efficiency of GAs in the context of combinatorial optimization problems[1] [3]. A combinatorial optimization problem can be mathematically described by a pair (S, f), where $S = \{s_1, s_2, ..., s_i, ...\}$ is a finite set of feasible solutions (a search space) and f: $S \to R$ is a real-valued objective (cost) function. We suppose that f seeks a global minimum. We also assume that solutions are represented by permutations of integer numbers from 1 to n, i.e. $S = \{s \mid s = (s(1), s(2), ..., s(n)), s(i) \in \{1, ..., n\}, i = 1, ..., n, s(i) \neq s(j), i, j = 1, ..., n, i \neq j\}$. Neighbouring solutions of the current solution can be determined using a neighbourhood function Θ: $S \to 2^S$ which assigns for each $s \in S$ its neighbourhood $\Theta(s) \subseteq S$. The solution $s^\bullet$ is said to be locally optimal with respect to the neighbourhood Θ if $f(s^\bullet) \leq f(s')$ for each $s' \in \Theta(s^\bullet)$. There may be lots of the locally optimal solutions over the search space. The goal is to find a solution $s^\diamond \in S$ such that $s^\diamond \in S^\diamond = \left\{ s^\nabla \mid s^\nabla = \arg\min_{s \in S} f(s) \right\}$; the solution $s^\diamond$ is called a globally optimal solution. Of course, a globally optimal solution is also a locally optimal solution with respect to any neighbourhood.

Many combinatorial optimization problems belong to the NP-hard class and finding globally optimal solutions for such problems within reasonable time limits may not be possible [1,16]. Therefore, heuristic methods like genetic algorithms are widely applied to obtain high-quality (near-optimal) solutions in moderate computation times.

The rest of this paper is structured as follows. In Section 1, the preliminaries and general aspects of enhanced-improvement-based genetic algorithms (EIGAs) are briefly discussed. Some variants of the enhanced improvement approach are concerned in Section 2. Section 3 describes the particular implementation of our approach for the quadratic assignment problem, as well as the results of the computational experiments. The paper is completed with concluding remarks.

1. Enhanced-improvement-based genetic algorithms: general aspects

Genetic algorithms are based on imitation of the natural process of evolution. Over generations, less fitted organisms fail to have offspring and tend to disappear, while more fitted individuals tend to predominate. Similarly, in the genetic algorithms, the goal

[1] As a typical example of the combinatorial optimization problem, a well-known problem, the quadratic assignment problem (QAP) [14] is considered.

is to arrive at high quality solutions by iteratively applying the standard evolution operations: selection, crossover (reproduction of offspring), mutation, and replacement (culling)[2].

In most cases, the landscapes (the trajectories of the values of the objective function) of hard combinatorial optimization problems (like the QAP) are extremely rugged with a huge number of local optima, which is in sharp contrast to the landscapes of simple unimodal problems (see Figure 1). In addition, good local optima are typically found in areas (clusters) distributed in a totally non-uniform way. All these circumstances cause severe difficulties for the local search-based heuristics, including the genetic algorithms. Indeed, the performance of the traditional GAs depends purely on the standard genetic operations, which are of the explorative nature, rather than improving (exploitative) operators. In order to achieve more efficiency, the additional improving algorithms (heuristics) are usually incorporated; at the same time, more attention is paid to the exploitative character of the genetic process [7, 22]. The resulting GAs are known as hybrid genetic (memetic) algorithms [20].

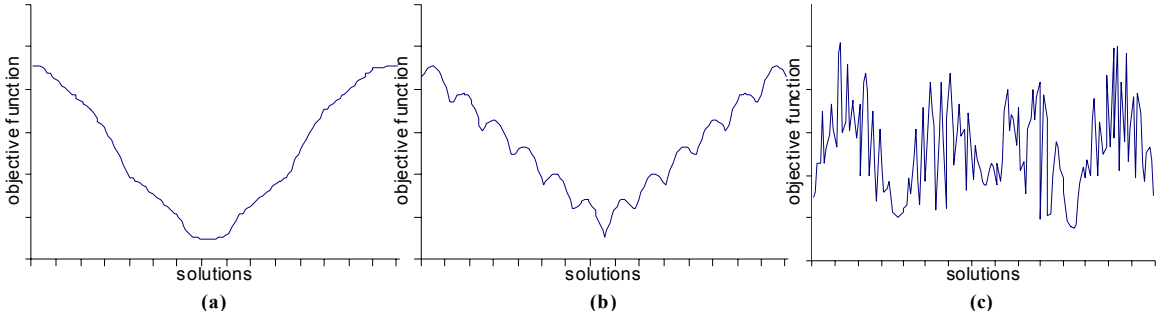

Figure 1. Illustrative examples of landscapes
((a) — simple unimodal problem, (b) — relatively easy problem with wide basin of attraction ("big valley"), (c) — hard optimization problem with isolated regions of local optima)

Although the hybrid genetic algorithms enable to get quite satisfactory solutions (see, for example, [8,10,17]), the quality of the obtained solutions may still be insufficient, especially when comparing to other refined intelligent optimization techniques like tabu search or simulated annealing. The main reason is the loss of the genetic variability of individuals followed by stagnation of the evolution process, as a result of a straightforward rapid improvement of offspring. Of course, there is always also a danger of falling into local optima without easy ways of escaping from them.

One of the alternatives to overcome these barriers is the development of new conceptual modifications of the hybrid GAs oriented to the exploitative ability of the genetic search. We propose a strategy called an "enhanced-improvement-based genetic algorithm" (EIGA). The central philosophy of this approach is to increase the effectiveness of the genetic search by concentrating, in particular, on the improvement of individuals during the evolution process. In contrast to the straightforward standard (hybrid) GAs that are based on evolution of primitive biological systems, the enhanced-improvement-based genetic algorithm rather imitates more complex, cultural environment, where the lifetime transformations and adaptations are very likely much more significant than the transmission of the parents' genetic information.

The most important features of EIGA are as follows.

A. EIGA operates with highly compact populations that consist solely of the superior quality individuals (super-individuals). This policy is completely different from that of classical GAs, which maintain quite large-sized populations with average or below-average individuals.

B. The super-individuals of EIGA can be obtained by means of powerful local-search based heuristics like tabu search, simulated annealing, iterated local search, etc. These heuristics might be quite time-consuming. However, the compensation could be achieved by utilizing very small-sized (miniature) populations — the size of populations is sacrificed for the increased computation time for the improvement of individuals.

C. The enhanced improvement of individuals needs an adequate diversification strategy to keep a balance between exploitative and explorative capabilities. The gentle mutations of the ordinary GAs would be clearly insufficient. Instead, more deep restructuring of the genotype of a population is desirable to prevent falling back into the previous (visited) local optima and drive the search to yet unexplored regions.

EIGA is a universal optimization methodology. There exists a great variety in the choice of how to design and implement the particular features (components) of the algorithm. Some possible variants of EIGA are briefly concerned in the next section.

[2] In genetic algorithms, the solutions are equivalent to individuals of a biological system and the cost of a solution (the value of the objective function) is equivalent to the fitness of an individual. For a more thorough description of the principles and applications of genetic algorithms, the reader is addressed to [2, 4, 6, 11, 12, 17, 21, 23-25, 29].

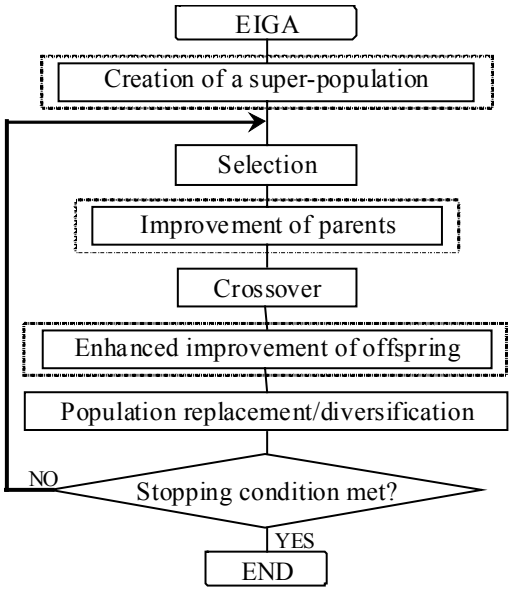

Figure 2. Generalized structure (flowchart) of the enhanced-improvement-based genetic algorithm

The generalized structure of EIGA is shown in Figure 2.

2. Variants of the enhanced-improvement-based genetic algorithm

2.1. Compounded approach

It is of extreme importance that EIGA starts with as good a population as possible. The compounded approach (CA) especially addresses this aspect. In the original version of CA [9], one starts with several populations (sub-populations), where the individuals of every population are independently optimized by the improving algorithm. After this, a pre-defined number of the best individuals are selected from these populations to form the single (compounded) initial population – similarly to migration of the best species to an elite population.

We may also use only one initial population instead of many sub-populations. We, however, have to use a "pre-improvement" (i.e. spend much more time by optimizing the members of the starting population). In this case, the number of generations of a genetic algorithm should be correspondingly decreased.

2.2. Quick improvement vs. time-expensive improvement

The enhanced genetic search is, in fact, based on the combination of the exploitative process (intensification) and explorative operations (diversification). Intensification (improving algorithm) aims at focusing the search within promising localized regions of the search space, while diversification (crossover, mutation) biases the search towards regions that are "far" from the current focus.

There may be different needs for the balance between intensification and diversification for different problems. This fact must be necessarily taken into consideration when designing EIGA for the particular problem (or even for the particular instance of the same problem).

If the degree of intensification, i.e. the number of improving iterations is small (this is the case of a quick (fast) improvement), then the convergence speed may possibly be slow. If it is large (this is the case of a time-expensive improvement), then the overall computational time increases. Fortunately, in the last case, the increased time for the over-intensified search can be effectively compensated by using more compacted populations. In addition, the number of generations may be accordingly decreased to keep the overall run time fixed.

2.3. Reinforced improvement

2.3.1. Improvement of parents

The underlying idea of this strategy is to apply the improving algorithm to the selected parents. The improvement is followed by the crossover procedure (of course, the improving algorithm is applied to the produced offspring, too). With the pre-crossover improvement, we are a bit closer to the nature — indeed, in the real life, only the best (young and healthy) members of a population are usually "licensed" to produce their offspring. The parent improvement just takes this point into account. Some variations of this approach are available; for example, it is possible that only one of the parents — the worse parent — undergoes the reinforced improvement. In this case, it is guessed that there probably is more potential to considerably improve the below-average individual than the above-average individual which is already of good

quality. In addition, doing so prevents the significant increase in run time of a genetic process.

2.3.2. Reinforced improvement of offspring

The other strategies are related to the enhanced improvement of offspring. For example, the following rule may be proposed. After producing and improving the offspring, it is tested if the new offspring is better than its parents. If this is not the case, the offspring is additionally improved by allotting a substantially increased number of improving iterations. (Otherwise, the algorithm continues in an ordinary way.) This seems to be a quite "altruistic" policy. It also means that some select individuals are given more chances than the rest of the "masses".

3. Computational experiments with the quadratic assignment problem

3.1. The quadratic assignment problem

In order to evaluate the efficiency of the proposed algorithm, the computational experiments have been carried out on the well-known combinatorial optimization problem, the quadratic assignment problem (QAP) [5,14]. This problem is formulated as follows. Given two matrices $\boldsymbol{A} = (a_{ij})_{n\times n}$ and $\boldsymbol{B} = (b_{kl})_{n\times n}$ and the set Π of permutations of the integers from 1 to n, find a permutation $\pi = (\pi(1), \pi(2), ..., \pi(n)) \in \Pi$ that minimizes

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

The QAP is a classical combinatorial optimization problem, where solutions are represented by permutations and the objective function is described according to the above formula. The QAP is used in many applications (including computer-aided design, factory/ office layout design, network design). It is thus a very good testing domain for different optimization methods.

3.2. Enhanced-improvement-based genetic algorithm and its variations for the QAP

The high-level description of the enhanced-improvement-based genetic algorithm for the QAP in a programming language-like form is presented in Figure 3.

```
procedure EnhancedImprovementBasedGeneticAlgorithm;
//input: n – problem size, A,B – flow and distance matrices, PS – population size, N_gen – # of generations,
//       N_offspr – # of offspring per generation, Q – # of improving iterations, W – tabu search depth, ρ – mutation rate
//output: π* – best solution found (resulting solution)
begin
  create the high-quality initial population P ⊂ Π (|P| = PS) in two steps:
   (1) generate members of P in a random way;
   (2) optimize each member of P by using the iterated tabu search algorithm IteratedTabuSearch;
  π* := argmin_{π∈P} z(π) ; //π* denotes the best so far solution
  for i := 1 to N_gen do begin
    P• := ∅;
    sort the members of P according to the ascending order of the values of the objective function;
    for j := 1 to N_offspr do begin //creation of the offspring
      select parents π', π'' ∈ P;
      apply the iterated tabu search algorithm IteratedTabuSearch to argmax{z(π'), z(π'')};
      apply crossover to π' and π'', get the offspring π''';
      improve the offspring π''' by using IteratedTabuSearch, get the improved solution π•;
      if z(π•) ≥ z(argmax{z(π'), z(π'')}) then apply reinforced improvement to π• by using IteratedTabuSearch;
      P• := P• ∪ {π•}; if z(π•) < z(π*) then π* := π• //saving the best so far solution
    endfor;
    remove N_offspr worst individuals from P ∪ P•, get the updated population P such that |P| = PS;
    if diversity lost then begin //population diversification
       (1) mutate the members of P by using the mutation procedure ControlledChainedMutation;
       (2) improve each mutated solution by using the improving procedure IteratedTabuSearch;
       if z(argmin_{π∈P} z(π)) < z(π*) then π* := argmin_{π∈P} z(π)
    endif
  endfor
end.
```

Figure 3. High-level pseudo-code of the enhanced-improvement-based genetic algorithm for the QAP

EIGA is initiated by creation of a fixed-size starting population $P = \{\pi_1, \pi_2, ..., \pi_{|P|}\} \subset \Pi$ ($|P| = PS$, where PS denotes the population size). Further, the selection, crossover, improvement, and replacement are applied iteratively until a stopping condition is satisfied.

For the construction of the initial population, we use the iterated tabu search (ITS) algorithm which has

been found to be greatly effective, in particular, for the quadratic assignment problem [19]. The pseudo-code of the ITS algorithm is given in Appendix, Figure A1. The ITS algorithm is, in turn, based on an improved robust tabu search (IRoTS) procedure and a special mutation procedure called a controlled chained mutation (CCM). IRoTS is similar to the robust tabu search algorithm due to Taillard [26], but has also several significant differences, among them, randomized tabu criterion, "intra-intensification" and "inter-intensification" mechanisms. The mutation procedure within the ITS algorithm is based on random pairwise interchanges of the elements (genes) of a solution (chromosome of an individual). The pseudo-codes of the IRoTS and CCM procedures are shown in Appendix, Figures A2, A3.

The functioning of the ITS algorithm is organized according to a so-called $(Q, W, 1, \rho)$-scheme. In this scheme, the number of improving iterations is equal to Q, whereas W is the tabu search depth (i.e. the number of iterations of the IRoTS procedure). Q may also be thought of as a measure of the search breadth (extensity), while W can be viewed as the search intensity. A single execution of the mutation procedure takes place every W iterations, and the mutation strength (disruptiveness) is defined by the parameter ρ.

The degree of intensification (exploitation) can be flexibly controlled, in particular, by the parameter Q. The decreased values of Q mean that the search is less intensified, whereas increasing the value of Q implies the enhanced improvement of the individuals.

In the compounded approach, we radically increase the number of improving iterations at the initialization phase, Q_{init}, to achieve higher quality initial population. In particular, $Q_{\text{init}} = 7Q$, where Q is the usual number of improving iterations used for the post-crossover improvement.

For the parents selection, a rank based selection rule [28] is applied. The offspring is produced by a cohesive crossover proposed by Drezner [8]. Note that, in our implementation of EIGA, we produce N_{offspr} children at each generation.

The improvement is performed by the use of the iterated tabu search algorithm, as mentioned above. By using time-expensive improvement, we increase the number of improving iterations by a factor of 2 (i.e. $Q_{\text{time\_expens}} = 2Q$). The parents are improved by using the standard number of improving iterations (Q), while, for the reinforced offspring improvement, we use a factor of 5 (i.e. $Q_{\text{reinforc}} = 5Q$) (this is due to the fact that the reinforced offspring improvement is, in general, more rare than the parent improvement).

During the replacement phase, the current population is updated by the new one. We apply an elitism strategy, that is, the individuals chosen for the next generation are the best PS members of $P_{PS} \cup P_{N_{\text{offspr}}}$, where P_{PS} is the population at the beginning of the current generation and $P_{N_{\text{offspr}}}$ denotes the set of newly created individuals (PS is the population size and N_{offspr} denotes the number of offspring per generation). In addition, if the diversity[3] of individuals of the obtained population is lost, then two auxiliary steps of the population diversification take place. In the first step, the members of the population are mutated in a quite aggressive manner (this is done using the CCM procedure with the maximally available mutation rate ($\rho = n$)). In the second step, the mutated solutions are transformed again into the optimized (elite) solutions. To get even more effect, we substantially increase the number of improving iterations during this step, Q_{diversif}; in particular, $Q_{\text{diversif}} = 5Q$, where Q is the default number of iterations.

The overall process is continued until a pre-defined number of generations, N_{gen}, have been performed.

The following short notations will be used for different variants of EIGA: EIGA^{CA} — compounded approach, EIGA^{TEI} – time expensive improvement of offspring, EIGA^{RI1} – reinforced improvement (parents only), EIGA^{RI2} – reinforced improvement (offspring only), EIGA^{RI3} – reinforced parent-offspring improvement, $\text{EIGA}^{\circledcirc}$ – combined variant ($\text{EIGA}^{\circledcirc} \equiv \text{EIGA}^{\text{CA}} \boxplus \text{EIGA}^{\text{TEI}} \boxplus \text{EIGA}^{\text{RI2}}$).

3.3. Results of computational experiments

We have examined our genetic algorithm on the benchmark problems taken from the quadratic assignment problem library – QAPLIB [5]. The following types of the QAP instances were tested:

a) uniform 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);

b) real-life like instances (they are designed to resemble real world problems (the distribution of the data is not uniform); these instances are denoted by tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, tai60b, tai80b, tai100b, and tai150b).

As a performance criterion for the algorithms, we use the average relative deviation ($\bar{\delta}$) of the solutions from the best known (pseudo-optimal) solution (BKS). It is defined by the formula: $\bar{\delta} = 100(\bar{z} - z^{\Diamond})/z^{\Diamond}\,[\%]$, where $\bar{z}$ is the average objective function value over 10 runs of the algorithm and $z^{\Diamond}$ denotes the best known value (BKV) of the objective function. (BKVs are from QAPLIB [5].)

[3] The diversity is measured by using the entropy of a population. The entropy of the population P is calculated according to the following formulas:

$$\text{Entropy}(P) = \sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij} \Big/ n\log_2 n\,,\quad e_{ij} = \begin{cases} 0, \kappa_{ij} = 0 \\ -\frac{\kappa_{ij}}{|P|}\log_2\frac{\kappa_{ij}}{|P|}, \text{otherwise} \end{cases},$$

where κ_{ij} is the number of times that the gene i occupies the locus j in the current population P (see also [18]).

The experiments were performed on a 1.8 GHz computer. The values of the main control parameters for different modifications of EIGA used in the experiments are collected in Table 1. The results of the comparison of different variants of EIGA ($EIGA^{CA}$, $EIGA^{TEI}$, $EIGA^{RI1}$, $EIGA^{RI2}$, $EIGA^{RI3}$, EIGA ⚙) are presented in Tables 2, 3.

Table 1. Main control parameters of EIGA

Parameter	Value		Remarks
	Random instances	Real-life like instances	
Population size, PS	$2\cdot\lfloor\sqrt{n}\rfloor$	$\lfloor\sqrt{n}\rfloor$	n is the size of the problem
Number of generations, N_{gen}	$n^{\ddagger}$	$10n$	
Number of offspring per generation, N_{offspr}	1	$PS/2$	
Number of improving iterations, Q	$5^{\ddagger\ddagger}$	5	
Search depth, W	n^2	n	
Mutation rate, ρ	$\lfloor 0.4n \rfloor$	$\lfloor 0.4n \rfloor$	

‡ in reinforced improvement and combined variants, the number of generations is correspondingly decreased;

‡‡ i) in $EIGA^{CA}$ and combined variant, the number of improving iterations at the initialization phase, Q_{init}, is equal to $7Q$;
ii) in $EIGA^{TEI}$ and combined variants, $Q_{time\_expens} = 2Q$;
iii) in reinforced offspring improvement mode and combined variant, $Q_{reinforc} = 5Q$;
iv) the number of improving iterations in the population diversification phase, $Q_{diversif}$, is equal to $5Q$.

Table 2. Results of the comparison of different variants of EIGA for the random QAP instances

Instance	N	BKV	$\bar{\delta}$						Average CPU time per run (sec.)
			$EIGA^{CA}$	$EIGA^{TEI}$	$EIGA^{RI1}$	$EIGA^{RI2}$	$EIGA^{RI3}$	EIGA ⚙	
tai20a	20	703482	**0.000**	**0.000**	0.030	**0.000**	**0.000**	**0.000**	*1.1*
tai25a	25	1167256	0.007	0.041	0.041	**0.000**	**0.000**	**0.000**	*2.6*
tai30a	30	1818146	0.010	0.009	**0.000**	**0.000**	**0.000**	**0.000**	*7.2*
tai35a	35	2422002	**0.000**	0.021	**0.000**	**0.000**	**0.000**	**0.000**	*16.5*
tai40a	40	3139370	0.194	0.245	0.222	0.263	0.213	**0.138**	*41*
tai50a	50	4938796	0.399	0.402	0.407	0.382	0.428	**0.363**	*140*
tai60a	60	7205962	0.498	0.524	0.412	**0.336**	0.394	0.343	*350*
tai80a	80	13515450	0.446	0.476	0.383	0.410	0.372	**0.371**	*1500*
tai100a	100	21054656	0.375	0.442	0.372	**0.315**	0.330	0.316	*5000*

Table 3. Results of the comparison of different variants of EIGA for the real-life like QAP instances

Instance	N	BKV	$\bar{\delta}$						Average CPU time per run (sec.)
			$EIGA^{CA}$	$EIGA^{TEI}$	$EIGA^{RI1}$	$EIGA^{RI2}$	$EIGA^{RI3}$	EIGA ⚙	
tai20b	20	122455319	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*0.04*
tai25b	25	344355646	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	*0.27*
tai30b	30	637117113	**0.000**	0.001	**0.000**	**0.000**	**0.000**	**0.000**	*0.58*
tai35b	35	283315445	**0.000**	0.003	**0.000**	0.007	0.005	**0.000**	*1.1*
tai40b	40	637250948	**0.000**	**0.000**	0.002	**0.000**	0.003	**0.000**	*2.3*
tai50b	50	458821517	**0.000**	0.009	0.010	0.007	0.011	0.001	*8.3*
tai60b	60	608215054	0.018	0.019	0.011	0.013	0.009	**0.003**	*11.8*
tai80b	80	818415043	0.009	0.025	0.034	0.022	0.023	**0.006**	*63*
tai100b	100	1185996137	0.048	0.099	0.083	0.067	0.072	**0.045**	*180*
tai150b	150	498896643	0.097	0.143	0.140	0.119	0.125	**0.055**	*900*

Overall, from Tables 2, 3, it could be viewed that all our algorithm variants are very effective and fast. We observed only slightly different behaviour of our algorithm variants for the uniform random and real-life like QAP instances. For example, it seems that, for the random instances, the reinforced improvement strategy yields quite promising results. Meanwhile, the compounded approach demonstrates very encouraging performance for the real-life like instances. Obviously, the combined enhanced variant is superior to the remaining variants; however, it consumes some more computational resources. It should be noted that the reinforced-offspring-improvement approach ($EIGA^{RI2}$) slightly outperforms both the parent-improvement and combined parent-offspring-improvement strategies ($EIGA^{RI1}$, $EIGA^{RI3}$). This means that the straightforward naive increasing of the improving iterations does not necessarily leads to the better performance.

We have also performed the comparisons of our enhanced genetic algorithm with other heuristic algorithms. The algorithms used in the comparison are as follows: robust tabu search (RoTS) [26], fast ant system (FANT) [27], simple genetic algorithm (SGA) [28], hybrid genetic-local search algorithm (HGLSA) [15], improved hybrid genetic algorithm (IHGA) [17], and enhanced-improvement-based genetic algorithm (combined version – EIGA ⚙). (SGA is a canonical genetic algorithm without local improvement. The next two GAs use local improvement procedures (descent local search – in HGLSA, tabu search – in

IHGA); however, the remaining structure of these algorithms (especially HGLSA) is quite different from that of EIGA.) The results of the comparison are shown in Table 4. They confirm the excellent efficiency of our approach when comparing with other methods.

Table 4. Results of the comparison of EIGA with other algorithms

Instance	n	BKV	$\overline{\delta}$						Average CPU time per run (sec.)
			RoTS	FANT	SGA	HGLSA	IHGA	EIGA ⚙	
tai20a	20	703482	0.071	1.011	7.034	0.197	**0.000**	**0.000**	*1.1*
tai20b	20	122455319	**0.000**	0.141	9.381	0.015	**0.000**	**0.000**	*0.04*
tai25a	25	1167256	0.153	1.543	7.008	0.500	0.007	**0.000**	*2.6*
tai25b	25	344355646	0.044	0.008	10.566	0.077	**0.000**	**0.000**	*0.27*
tai30a	30	1818146	0.068	1.100	6.998	0.905	**0.000**	**0.000**	*7.2*
tai30b	30	637117113	0.488	0.043	13.982	0.456	**0.000**	**0.000**	*0.58*
tai35a	35	2422002	0.222	1.321	6.943	1.245	0.005	**0.000**	*16.5*
tai35b	35	283315445	0.288	0.233	14.006	0.279	0.002	**0.000**	*1.1*
tai40a	40	3139370	0.484	1.545	6.921	1.463	0.219	**0.138**	*41*
tai40b	40	637250948	0.208	**0.000**	12.002	0.195	0.001	**0.000**	*2.3*
tai50a	50	4938796	0.789	1.882	7.407	1.583	0.428	**0.363**	*140*
tai50b	50	458821517	0.253	0.230	11.065	0.369	0.008	**0.001**	*8.3*
tai60a	60	7205962	0.801	1.824	8.417	1.736	0.598	**0.343**	*350*
tai60b	60	608215054	0.294	0.179	10.592	0.418	0.009	**0.003**	*11.8*
tai80a	80	13515450	0.756	1.436	9.383	1.156	0.422	**0.371**	*1500*
tai80b	80	818415043	0.290	0.325	14.734	0.448	0.019	**0.006**	*63*
tai100a	100	21054656	0.694	1.376	10.987	1.320	0.389	**0.316**	*5000*
tai100b	100	1185996137	0.197	0.099	15.086	0.208	**0.040**	0.045	*180*
tai150b	150	498896643	0.399	0.554	17.378	0.423	0.056	**0.055**	*900*

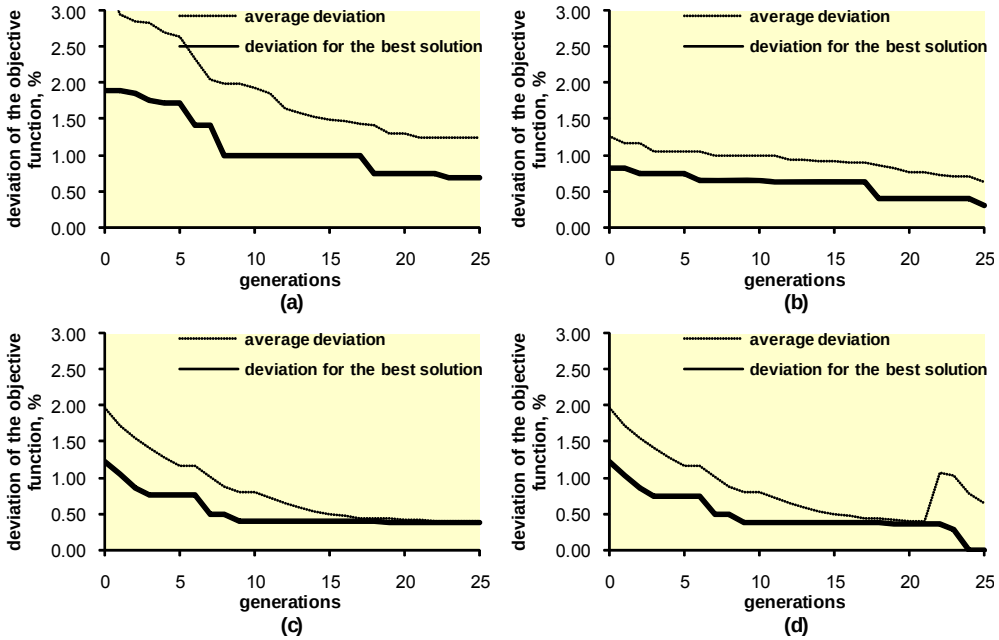

Figure 4. Graphical illustrations of the behaviour of different variants of EIGA for the QAP instance tai30a: (a) basic variant (no pre-improvement, no enhanced improvement, no diversification), (b) compounded approach, (c) enhanced (time-expensive) improvement, (d) enhanced improvement combined with population diversification (Note. Average deviation for the entire population and deviation for the best member of the population are given)

In Figure 4, the behaviour of different variants of EIGA for the QAP instance tai30a is graphically depicted. From this figure, the impact of the enhanced improvement on the quality of solutions is clearly visible. Note that, in compounded approach, one starts from the better quality population, however the process converges less rapidly (see Figure 4b). The effect of population diversification can also be seen (see Figure 4d). Also notice that, in Figure 4d, the increase in the average deviation just corresponds to the moment of population diversification. We have observed a similar kind of behaviour for all other QAP instances.

4. Concluding remarks

In this paper, a new conceptual modification of the genetic algorithms entitled as an enhanced-improvement-based genetic algorithm (EIGA) is proposed.

The key idea of EIGA is to speed up the convergence of the evolutionary process by operating with the outstanding quality individuals (super-individuals). The enhanced improvement of individuals is coupled with strong enough mutation to keep a proper balance between exploitative and explorative capabilities of the genetic algorithm. Miniature populations are used to compensate the increase in time for the improvement of individuals.

Our genetic algorithm is implemented and tested on the hard combinatorial optimization problem, the quadratic assignment problem. The results obtained from the experiments with the random and real-life like QAP instances show great potential of using the superior-quality individuals in a genetic algorithm.

The further investigations of the enhanced-improvement-based approach would be worthwhile. Among other things, it would be worthy to analyze the impact of using different kind of diversification strategies within the enhanced-improvement-based GAs. It might also be worthy to apply this type of genetic search to other combinatorial optimization problems.

References

[1] **E.H.L. Aarts, J.K. Lenstra (eds.).** Local Search in Combinatorial Optimization, *Wiley, Chichester*, 1997.

[2] **J.E. Beasley, P.C. Chu.** A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 1996, *Vol*.94, 392–404.

[3] **C. Blum, A. Roli.** Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, 2003, *Vol*.35, 268–308.

[4] **T.N. Bui, B.R. Moon.** Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 1996, *Vol*.45, 841–855.

[5] **R.E. Burkard, S. Karisch, F. Rendl.** QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 1997, *Vol*.10, 391–403 (*http://www.seas.upenn.edu/qaplib, cited* 14 *June* 2008).

[6] **R. Chelouah, P. Siarry.** A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics*, 2000, *Vol*.6, 191–213.

[7] **D. Costa, A. Hertz, O. Dubuis.** Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics*, 1995, *Vol*.1, 105–128.

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

[9] **Z. Drezner.** Compounded genetic algorithms for the quadratic assignment problem. *Operations Research Letters*, 2005, *Vol*.33, 475–480.

[10] **B. Freisleben, P. Merz.** A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *In T.Bäck, H.Kitano, Z.Michalewicz (eds.)*, *Proceedings of the* 1996 *IEEE International Conference on Evolutionary Computation*, *IEEE Press, New York*, 1996, 616–621.

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

[12] **R.L. Haupt, S.E. Haupt.** Practical Genetic Algorithms. *Wiley-Interscience, Hoboken, NJ*, 2004.

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

[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] **W. Michiels, E.H.L. Aarts, J. Korst.** Theoretical Aspects of Local Search. *Springer, Berlin-Heidelberg*, 2007.

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

[18] **A. Misevičius.** An entropy-based genetic algorithm. *In L.Sakalauskas, G.W.Weber, E.K.Zavadskas (eds.), Proceedings of the* 20*th International Conference, EURO Mini Conference "Continuous Optimization and Knowledge-Based Technologies" (EurOPT'* 2008*), VGTU Publishing House "Technika", Vilnius*, 2008, 7–12.

[19] **A. Misevičius, A. Lenkevičius, D. Rubliauskas.** Iterated tabu search: an improvement to standard tabu search. *Information Technology and Control*, 2006, *Vol*.35, *No*.3, 187–197.

[20] **P. Moscato.** Memetic algorithms. *In P.M.Pardalos, M.G.C.Resende (eds.), Handbook of Applied Optimization, Oxford University Press, New York,* 2002, 157–167.

[21] **M. Paulinas, A. Ušinskas.** A survey of genetic algorithms applications for image enhancement and segmentation. *Information Technology and Control*, 2007, *Vol*.36, *No*.3, 278–284.

[22] **C.R. Reeves, C. Höhn.** Integrating local search into genetic algorithms. *In V.J.Rayward-Smith, I.H.Osman, C.R.Reeves, G.D.Smith (eds.), Modern Heuristic Search Methods, Wiley, Chichester*, 1996, 99–115.

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

[24] **M.Senthil Kumar.** Genetic algorithm-based proportional derivative controller for the development of active suspension system. *Information Technology and Control*, 2007, *Vol*.36, *No*.1, 58–67.

[25] **S.N. Sivanandam, S.N. Deepa.** Introduction to Genetic Algorithms. *Springer,Berlin-Heidelberg-New York*, 2008.

[26] **E. Taillard.** Robust taboo search for the QAP. *Parallel Computing*, 1991, *Vol*.17, 443–455.

[27] **E. Taillard.** FANT: fast ant system. *Tech. Report IDSIA-46-98, Lugano, Switzerland*, 1998.

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

[29] **T. Yamada, R. Nakano.** A genetic algorithm applicable to large-scale job-shop problems. *In R.Männer, B.Manderick (eds.), Parallel Problem Solving from Nature* 2, *North-Holland, Amsterdam*, 1992, 281–290.

Received July 2008.