ISSN 1392 – 124X INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS, 2004, Nr. 4(33)

AN EXTENSION OF HYBRID GENETIC ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM[♣]

Alfonsas Misevičius

Kaunas University of Technology, Department of Practical Informatics,
Studentų st. 50–400a/416a, LT–51368 Kaunas, Lithuania

Abstract. Genetic algorithms (GAs) are modern population based heuristic approaches. Recently, GAs have become very popular by solving various optimization problems. In this paper, we discuss an extension of a hybrid genetic algorithm for the well-known combinatorial optimization problem, the quadratic assignment problem. This extension is based on a promising genetic-tabu search policy. An enhanced tabu search is used in the role of the local improvement of solutions, whereas a robust mutation (reconstruction) strategy is "responsible" for maintaining a high degree of the diversity within the population and for avoiding a premature convergence of GA. We tested our algorithm on a set of the QAP instances. The results obtained show the outstanding performance of the proposed algorithm.

Keywords: combinatorial optimization, heuristic algorithms, genetic algorithms, tabu search, quadratic assignment problem.

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 the permutations of the integers from 1 to n be given. The goal is to find a permutation $\pi = (\pi(1), \pi(2), ..., \pi(n)) \in \Pi$ that minimizes

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

One of the interpretations of the QAP is that of Koopmans and Beckmann [14]. In this case, one deals with locating n facilities on n locations with some physical products flowing between the facilities, and with distances between the locations. The element a_{ij} is the flow from the facility i to facility j, and the element b_{kl} is the distance between the locations k and l. The permutation $\pi = (\pi(1), \pi(2), ..., \pi(n))$ represents an assignment of facilities to locations (here, $\pi(i)$ ($\pi(i) \in \{1, 2, ..., n\}$) denotes the location facility i is assigned to). Solving the QAP means searching for an assignment that minimizes the "transportation cost" between facilities, z.

It has been proved that the QAP is NP-hard [23], therefore heuristic approaches [18] are used for solving medium- and large-scale QAPs in reasonable times, among them, ant algorithms [8], greedy randomized adaptive search procedures [15], iterated local search [24], scatter search [4], simulated annealing [2], tabu search [20,25]. Starting from 1994, several authors applied the genetic algorithms to the QAP, first of all [1,6,7,16,17,19,28].

The QAP is a representative (instance) of combinatorial optimization (CO). CO may be described in the following way. Let S be the set of (feasible) solutions (the solution space); furthermore, let f: $S\rightarrow\Re$ be the objective function (we presume that f seeks a global minimum). Solving an instance of CO problem (S, f) means searching for a solution $s_{\text{opt}} \in S$ such that

$$s_{\text{opt}} \in S_{\text{opt}} = \left\{ s^{\nabla} \mid s^{\nabla} = \arg\min_{s\in S} f(s) \right\}. \qquad (2)$$

The solution s_{opt} is called a globally optimal solution (global optimum), and the set $S_{\text{opt}} \subseteq S$ denotes the set of global optima. In addition, a neighbourhood function Θ: $S\rightarrow 2^S$ may be defined. It attaches for each $s\in S$ a set $\Theta(s)\subseteq S$ – a set of neighbouring solutions of s. Each solution $s' \in \Theta(s)$ can be reached from s by an operation called a move, and s is said to move to s' when such an operation is performed. The 2-exchange neighbourhood function, Θ_2, is widely used in the environments of permutation-based solutions, like the QAP. For the solution s, Θ_2 gives the set $\Theta_2(s) = \{s' \mid s' \in S,\ \rho(s,s') = 2\}$, where $\rho(s,s')$ is the "distance" between solutions s and s'. The natural way

[♣] This work is supported by Lithuanian State Science and Studies Foundation through grant number T-04078.

of defining the "distance" between solutions-permutations is counting the items that are assigned to different positions of the solutions, i.e.

$$\rho(s,s') = \left|\{i \mid s(i) \neq s'(i)\}\right| . \qquad (3)$$

We will use the compact notation m_{ij} ($i, j = 1, 2, ..., n$) for the move from s to $s' \in \Theta_2(s)$, which exchanges ith and jth elements in the solution s to get s'. (In this case, the expression $s' = s \oplus m_{ij}$ means that s' is obtained from s by applying m_{ij}.)

The remaining part of this paper is organized as follows. In Section 1, the principles of standard and hybrid genetic algorithms are outlined. The new proposed extension of the hybrid genetic algorithm for the quadratic assignment problem is discussed in Section 2. In Section 3, we present computational results obtained by examining some instances of the QAP. Finally, Section 4 completes the paper with concluding remarks.

1. Genetic algorithms

1.1. Standard genetic algorithms

The original concepts of genetic algorithms (GAs), which are based on the biological process of natural selection, were developed by Holland [13] in 1975. Genetic algorithm operates with a group P (called a population) of solutions $s_1, s_2, ..., s_{PS}$ (called individuals) from S. Each individual (s_i) is associated with some fitness corresponding to the objective function value ($f(s_i)$). In the case of minimization problem, the less the objective function value, the more fitting the individual, and the larger is the probability that the individual will survive in evolution process. During many generations, best fitting individuals tend to dominate, while less fitting ones tend to die off.

The general framework for the standard genetic algorithm can be described in the following way. A fraction of P is chosen to be parents by use of the selection function (it can formally be defined as a mapping ϕ: $2^S \rightarrow S \times S$). New individuals (i.e. offspring) are created by combining the information contained in the parents (this recombination operator is known as a crossover, ψ: $S \times S \rightarrow S$). Afterward, some members of the population undergo random perturbations (called mutations, ζ: $S \rightarrow S$) to prevent a premature loss of the individuals' diversity within the population. Finally, a replacement (culling) scheme, φ: $2^S \rightarrow 2^S$, is applied to determine which individuals survive to form the next generation. This process is to be repeated until some termination criterion is met.

There exists a great variety in the choice of the particular selection, crossover, mutation, and other related procedures. The detailed material on this topic, as well as the foundations and applications of GAs can be found in [5, 12, 22].

1.2. Hybrid genetic algorithms

Usually, the results obtained by the "pure" genetic algorithms are of rather poor quality. This fact was a motivation to embed additional heuristic components into the standard GAs. The examples of such components may be: a) including the construction heuristics for generation the initial populations; b) designing the special heuristic crossover (recombination) operators tailored to the specific characteristics of the problem; c) incorporating the local search heuristics to be applied to the solutions built by the crossover operator. Other enhancements are possible, for example, using a so-called restart mechanism in the cases of a loss of the diversity. These additional components (features), plus the standard genetic algorithm are what one calls a hybrid genetic (memetic) algorithm (HGA) [21].

Very roughly, the typical steps of HGA are as follows. The first step is to create an improved initial population by means of the known constructive and/or local search algorithms. The result of this step is a population which represents a collection of locally optimal solutions. After the creation of the optimized initial population, the standard procedures take place. Like in the standard GAs, the selected individuals undergo the crossover for creating new individuals. The important feature of HGA is that one operates with the optimized solutions as the inputs to the crossover operator. Although the crossover operator is highly "responsible" for the efficiency of genetic algorithms, many researchers came to the conclusion that it is insufficient to achieve competitive performance. The reason is that the offspring produced by the crossover is in general not locally optimal. The way out is just incorporating a post-crossover (local improvement) procedure to be applied to each offspring to obtain again locally optimal solutions.

The mutation is then performed on the locally optimized offspring. Regarding the mutations, the following should be said. The solution perturbed by the mutation operator is again transformed into an optimized solution to keep the local optimality of the population; so, the mutations are highly desirable to be strong enough to minimize the possibility of a possible falling back into previous local optima.

The population replacement scheme within HGA is also specific. It must guarantee a sufficient degree of the diversity of the population, which is very important by avoiding a premature convergence of GA.

To conclude, the hybridization of GA is an essential improvement over the standard GA. This is mainly due to the fact that, in HGA, the population solely of local optima is maintained. So, we can view the hybrid genetic search as the search over an optimized, high quality solution space — this appears to be much more effective process than when searching in a random (or slightly improved) solution space.

2. An extension of hybrid genetic algorithm for the QAP

In this section, we describe an extended hybrid genetic algorithm (EHGA) for the quadratic assignment problem. The algorithm starts with the creation of an initial population P. This is done in two steps: firstly, $PS=|P|$ permutations are generated in a pure random way; secondly, all the individuals of the population just produced are improved by a local search. Eventually, the population members are sorted according to the increasing values of the objective function. A tabu search (TS) [9,10,11] based algorithm, namely a so-called enhanced tabu search (ETS) procedure is used in the role of the local improvement technique (for the details of ETS, see below). So, the input for the further genetic operators is a population that consists of PS locally optimal solutions.

The algorithm then proceeds in the following way. Two solutions are selected to be parents of a new individual (child). For the parents selection, we apply a rank based selection rule [28]. The position, u, of the parent within the sorted population is determined by to the formula $u=\lfloor v^{\sigma}\rfloor$, where v is a uniform random number from the interval $[1,\ PS^{1/\sigma}]$, where PS is the population size, and σ is a real number in the interval $[1, 2]$ (it is referred to as a selection factor). It is obvious that the better the individual, the larger probability of selecting it for the crossover.

For the parents merging, we use a variant of the crossover operator proposed in [28] (it is entitled as a uniform like crossover (ULX)). ULX works as follows. First, all items assigned to the same position in both parents are copied to this position in the child (i.e. $c_i=a_i=b_i$, where c_i, a_i, b_i are the values in the i-th position of the permutations-parents and the permutation-child, respectively). 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 (i.e. $c_i=\begin{cases} a_i, r<0.5 \\ b_i, \text{otherwise}\end{cases}$, where r is a random number within the interval [0,1]). Third, remaining items are assigned at random (this step is needed to preserve the feasibility of the resulting permutation). The ULX operator implies a high degree of randomness. Even the same pair of parents may produce lots of quite different children, especially, when the "distance" (see formula (3)) between parents is large. So, we can extend the functioning of the crossover by making it create "m-plets" ($m>1$, typically $m=O(n)$) instead of a single child. Some kind of tournament among m pretenders takes place to determine the best candidate for survival. The "winner", usually the child which has the smallest objective function value is the only output of the crossover (see Figure 1). We call this type of proceeding an elitist crossover (EX). The search could often be improved even more if the crossover is applied more than once at the same generation. In our implementation, the number of EXs per one generation is controlled by the parameter N_{cross} (as a rule, the value of N_{cross} depends on the population size).

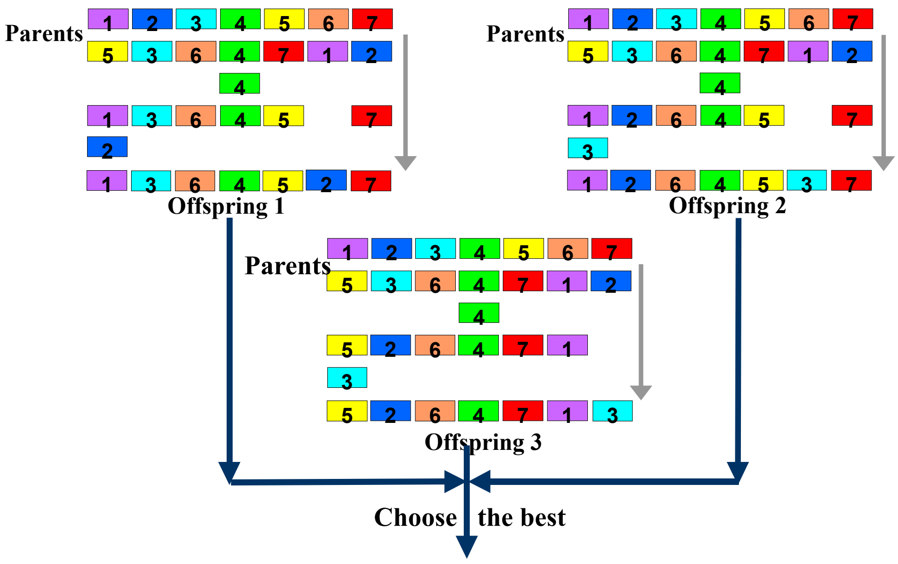

Figure 1. Example of producing a child in the "triplet" crossover

As mentioned in Section 1.2, genetic algorithms succeed in search if only they dispose of a robust local search based procedure. Ideally, such a procedure should be both fast and produce good quality solutions. Practically, these features hardly "intersect". A quite good candidate to meet the conflicting requirements is the tabu search, the method which has been proven to be extremely effective [11]. This is especially true for the quadratic assignment problem. So, we have chosen this approach as a perfect compromise between the opposites mentioned. Namely, we use the enhanced tabu search algorithm [20], a quite promising approach for the QAP. ETS plays the central role in our hybrid genetic algorithm. The idea behind ETS is that the classical TS (it serves as an intensification mechanism) is combined with the appropriate perturbations of solutions (as a robust diversification mechanism). The intensification itself is based on the robust tabu search procedure due to Taillard [25]. Very roughly, this algorithm can be outlined in the following way. Initialize the tabu list ***T***, and start from the current permutation π. Then, continue the following process until a predetermined number of steps, τ, have been performed:

a) find a neighbour π'' of the current solution π in such a way that $\pi'' = \arg\min_{\pi' \in N_2'(\pi)} z(\pi')$, where $N_2'(\pi) = \{\hat{\pi} \mid \hat{\pi} \in N_2(\pi),\ \hat{\pi} = \pi \oplus m_{ij}$ and $((m_{ij}$ is not tabu) or $(f(\hat{\pi}) < f(\pi^*)))\}$ where π^* is the best so far solution;

b) update the tabu list ***T*** by including the move m_{uv}, where m_{uv} is the move from the solution π to the solution π'';

c) replace the current permutation π by the neighbour π'', and use as a starting solution for the next step.

In EHGA, we usually apply the short runs of the tabu search (we call this strategy a limited tabu search (LTS)). Firstly, LTS allows saving the computation time; on the other hand, LTS in combination with other genetic operators is quite enough to seek for high quality solutions.

Regarding the diversification mechanism, we use the permutation mutations which can be seen as sequences (strings) of random moves $m_{r_1r_2}, m_{r_3r_4}, \ldots, m_{r_{2\mu-1}r_{2\mu}}$. The larger the length of the sequence (i.e. the mutation level) μ, the stronger the mutation, and vice versa. In turn, the stronger the mutation, the more the probability that the "distance" ρ between the current solution and the mutated one is also large. One can add more robustness to the diversification process by performing "concentric" mutations. In this case, the mutation level μ varies as follows: at the beginning, μ is equal to some minimum value $\mu_{\min}$; further, μ is increased gradually, step by step, until some limit is reached (this means that the new produced solutions are more and more "far" from an imaginary "center-solution"); once the maximum level $\mu_{\max}$ has been reached (or a better local optimum has been found), the current value of μ is immediately dropped to $\mu_{\min}$, and so on.

Note that, as the solutions already undergo perturbations in the ETS procedure, there is no need in any mutations within GA itself (except the special case discussed below).

As to the way in which the candidates for the subsequent mutation are chosen, a so-called exploration strategy is applied. The idea of exploration is that every new locally optimal solution (obtained by the intensification procedure), no matter its quality, is accepted for the reconstruction. The advantage of this strategy is allowing to search in many possibly promising regions of the solution space.

Some remarks on the way the combination of intensification and diversification is done should be mentioned. We rely upon so-called $(Q,\tau,1)$-scheme. In this scheme, the total number of the iterations of ETS (i.e. global iterations) is equal to Q. At each global iteration, τ steps (local iterations) of TS, and one call to the mutation procedure are performed. The value of aspect ratio Q/τ is of high importance. It may be viewed as a measure for the mutation frequency. (Suppose, $Q\tau$=const; in this case, the larger the value of Q/τ, the larger the mutation frequency, and vice versa.) The optimal value of Q/τ can only be revealed empirically. Some results of the experiments on the autonomous runs of ETS show that this value depends on the nature of the problem being solved. For random data (n=40), we found that the optimal value is somewhere between 0.05 and 5; for real world data, the situation is different: the higher the mutation frequency, the better the results (see Figure 2).

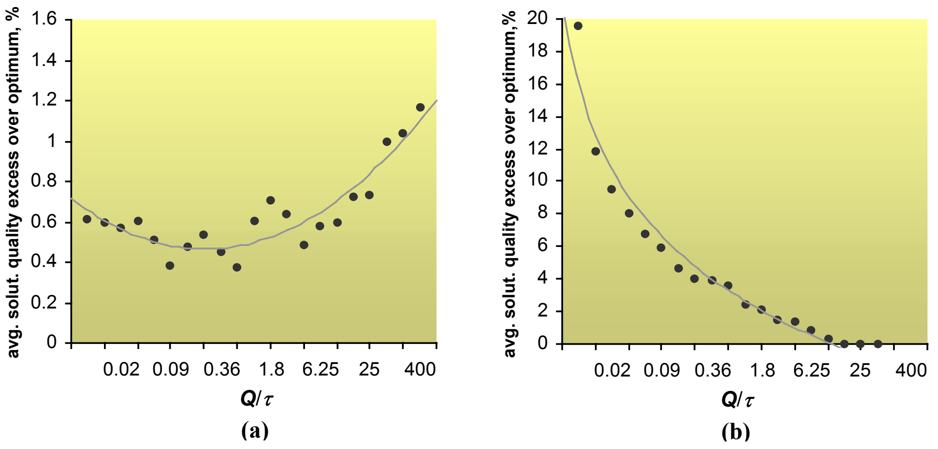

Figure 2. Solution quality (objective function value) versus the mutation frequency: a) random problem, b) real world problem

The culling (replacement) of the population takes place every time before going to the next generation. After adding new solutions and sorting the population, the solutions with the greatest objective function value are removed away from the population to keep the population size constant. After that, EHGA proceeds with the new population as described above. An extra dealing with the population (i.e. a so-called restart mechanism) is applied under some circumstances. That is, if the situation of a loss of the diversity of the population is identified, then a specific process (a "cold restart") is invoked. (A measure of entropy of the population [7] seems to be an applicable restart criterion; for more details, see [19].) In fact, there are two phases at the restart: a) the deep reconstruction (mutation) of all the members of the population; b) the local improvement of the reconstructed solutions by the limited enhanced tabu search. For the reconstruction, we tried the procedure very similar to that used in the ETS algorithm. The main difference lies in the increased mutation level; more precisely, the reconstruction is done in such a way that $\rho(\pi,\tilde{\pi}) = n$, where $\pi,\tilde{\pi}$ are the solutions before and after reconstruction; n is the problem size. After restart, EHGA goes on in an ordinary way. The overall process is continued in an iterative way until a given number of generations (N_{gen}) has been performed.

The generalized templates (frameworks) of the extended hybrid genetic algorithm, as well as the enhanced tabu search algorithm are presented in Figures 3 and 4.

```
procedure ExtendedHybridGeneticAlgorithm;
   // input: $A,B$ – the flow and distance matrices, $n$ – the problem size; output: $\pi^*$ – the best permutation found //
   // parameters: $PS$ – the population size, $N_{gen}$ – # of generations, $\sigma$ – the selection factor, //
   //              $N_{cross}$ – # of crossovers per generation, $Q$ – # of iterations of ETS, //
   //              $\tau$ – # of intensification iterations, $\alpha_1,\alpha_2$ – the mutation factors //
   $\mu_{min} := \max(2,\lfloor\alpha_1 n\rfloor)$; $\mu_{max} := \max(2,\lfloor\alpha_2 n\rfloor)$;
   create the locally optimized population $P\subset\Pi$ in two steps:
    (i) generate initials solutions of $P$ randomly,
   (ii) improve each member of $P$ by using the (limited) enhanced tabu search;
   $\pi^* := \operatorname{argmin}_{\pi\in P} z(\pi)$ ; // $\pi^*$ is the best so far solution //
   for $i := 1$ to $N_{gen}$ do begin // main cycle of the extended hybrid genetic algorithm //
     sort the members of $P$ in the ascending order of their fitness;
     for $j := 1$ to $N_{cross}$ do begin // in this cycle, $N_{cross}$ children will be produced //
       select parents $\pi',\pi'' \in P$ ;
       apply elitist crossover EX to $\pi'$ and $\pi''$, get the offspring $\dot{\pi}$ ;
       $\pi^\bullet :=$ EnhancedTabuSearch($\dot{\pi}$); // every offspring is improved by applying $Q$ iterations of ETS //
       add the improved permutation $\pi^\bullet$ to the population $P$;
       if $z(\pi^\bullet) < z(\pi^*)$ then $\pi^* := \pi^\bullet$ // save the best so far solution (as a possible result of EHGA) //
     end; // for $j$ ... //
     cull the population $P$ by removing $N_{cross}$ worst individuals;
     if the diversity of $P$ is below the predefined threshold then
        make the "restart" in two phases:
          (i) reconstruct (mutate) all the members of $P$, except the best one,
         (ii) improve each reconstructed solution by using
              the limited enhanced tabu search
              (save the new best encountered solution if any)
   end; // for $i$ ... //
end.
```

Figure 3. Template of the extended hybrid genetic algorithm

```
function EnhancedTabuSearch($\pi$);
  // input: $\pi$ – the current permutation; output: $\pi^*$ – the best permutation found //
  // parameters: $Q$, $\tau$, $\mu_{min}$, $\mu_{max}$ //
  apply $\tau$ iterations of (standard) robust tabu search to $\pi$,
  get the improved permutation $\pi^\bullet$;
  $\pi := \pi^\bullet$; $\pi^* := \pi^\bullet$; $\mu := \mu_{min} - 1$;
  for $q$ :=1 to $Q$ do begin // main cycle of the enhanced tabu search //
    accept candidate $\pi$ for the subsequent mutation (perturbation);
    if $\mu < \mu_{max}$ then $\mu := \mu + 1$ else $\mu := \mu_{min}$; // update the mutation level //
    apply mutation to $\pi$ with the mutation level $\mu$,
    get the new permutation $\tilde{\pi}$;
    apply $\tau$ iterations of (standard) robust tabu search to $\tilde{\pi}$,
    get the new improved permutation $\pi^\bullet$;
    if $z(\pi^\bullet) < z(\pi^*)$ then begin // new locally optimal solution is found //
      $\pi^* := \pi^\bullet$; // save the best so far solution (as a possible result of ETS) //
      reset the mutation level $\mu$
    end
  end; // for //
  return $\pi^*$
end.
```

Figure 4. Template of the enhanced tabu search

3. Computational results

In order to evaluate the performance of the proposed algorithm, the computational experiments have been carried out on the QAP instances taken from the well-known publicly available library of the QAP instances QAPLIB [3]. The classes 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, tai100a);

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

We compared our algorithm with other five different heuristic algorithms. The following algorithms were used: 1) robust tabu search (RTS) algorithm [25]; 2) fast ant system (FANT) [26]; 3) genetic hybrid (GH) algorithm [7]; 4) genetic algorithm due to Lim, Yuan and Omatu (GA-LYO) [16]; 5) improved hybrid genetic algorithm (IHGA) [19]. Note that RTS is among the best heuristic algorithms for the random instances, whereas IHGA belongs to the most powerful algorithms for the real-life like problems. The performance measures are as follows: a) the average deviation 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 the algorithm to a given instance), and $\breve{z}$ is the best known value (BKV) of the objective function); b) the number of solutions that are within 1% optimality (over 10 restarts) – $C_{1\%}$; c) the number of the best known values (solutions) found – C_{bkv}.

The values of the control parameters of the algorithms were chosen in such a way that all the algorithms use approximately the same computation (CPU) time. The main parameter values of EHGA are collected in Table 1.

Note that the quite different values of the parameter τ were chosen for the problem types (a) and (b). This is due to the fact that, for the random instances, more attention should be given (i.e. more time should be allotted) to the intensification; whereas, for the real-life like instances, the intensification is relatively less important than the diversification (see also [20]). Consequently, for the random instances, the parameter τ gets larger values than for the real-life like instances.

The results of the experiments are presented in Tables 2 and 3.

Table 1. Values of the parameters for the algorithm EHGA

Problem type	PS	N_{gen}	σ	N_{cross}	Q	τ	α_1	α_2
(a)	$\sqrt{n}$	$\frac{1}{4}n$	1.3	$\frac{1}{4}PS$	5	$\frac{1}{2}n^2$	depends on problem[a]	depends on problem[b]
(b)	$\sqrt{n}$	depends on problem[c]	1.7	$\frac{1}{2}PS$	5	n	depends on problem[a]	depends on problem[b]

[a] varies from 0.2 to 0.3; [b] varies from 0.3 to 0.4; [c] varies from $\frac{1}{4}n$ to $\frac{1}{2}n$.

Table 2. Comparison of the algorithms on randomly generated 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	n	BKV	$\bar{\delta}$, $C_{1\%}/C_{bkv}$												CPU time
			RTS		FANT		GH		GA-LYO		IHGA		EHGA		
tai20a	20	703482 [a]	0.07	10/6	0.90	1/0	0.41	10/2	1.30	3/0	0.06	10/8	**0**		*1.3*
tai25a	25	1167256 [a]	0.14	10/6	1.34	5/1	0.39	10/2	1.54	1/0	0.08	10/7	**0**		*4.4*
tai30a	30	1818146 [a]	0.08	10/5	1.11	4/1	0.37	10/4	1.56	1/0	0.02	10/8	**0**		*10.0*
tai35a	35	2422002 [a]	0.19	9/3	1.28	3/0	0.64	10/0	1.60	0/0	0.05	10/7	**0**		*36*
tai40a	40	3139370 [a]	0.46	8/0	1.55	2/0	0.62	9/0	1.95	0/0	0.21	10/1	**0.20**	10/1	*85*
tai50a	50	4941410 [a]	0.79	5/0	1.76	1/0	0.87	4/0	2.01	0/0	0.44	9/1	**0.43**	10/1	*290*
tai60a	60	7205962 [b]	0.84	2/0	1.71	0/0	1.01	2/0	1.93	0/0	**0.55**	8/0	0.56	8/0	*720*
tai80a	80	13546960 [b]	0.62	7/0	1.34	5/0	0.60	6/0	1.30	1/0	0.32	10/0	**0.27**	10/0	*3300*
tai100a	100	21123042 [b]	0.59	8/0	1.19	6/0	0.51	8/0	1.16	2/0	0.26	10/0	**0.23**	10/0	*12000*

[a] comes from [3]; [b] comes from [20].

Table 3. Comparison of the algorithms on 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	n	BKV	$\bar{\delta}$, $C_{1\%}/C_{bkv}$												CPU time
			RTS		FANT		GH		GA-LYO		IHGA		EHGA		
tai20b	20	122455319 [a]	**0**		0.09	10/8	0.05	10/9	0.10	10/7	**0**		**0**		*0.1*
tai25b	25	344355646 [a]	0.06	10/8	0.01	10/9	**0**		0.01	19/8	**0**		**0**		*0.6*
tai30b	30	637117113 [a]	0.40	9/3	0.04	10/7	0.01	10/9	0.50	9/1	**0**		**0**		*1.2*
tai35b	35	283315445 [a]	0.25	10/5	0.20	10/1	0.13	10/4	0.27	10/0	0.00	10/9	**0**		*2.5*
tai40b	40	637250948 [a]	0.20	9/6	0.01	10/9	**0**		0.60	8/0	**0**		**0**		*5.0*
tai50b	50	458821517 [a]	0.24	10/0	0.22	9/0	0.03	10/7	0.95	5/0	0.02	10/8	**0**		*18*
tai60b	60	608215054 [a]	0.30	10/0	0.18	9/3	0.02	10/6	0.80	4/0	0.01	10/9	**0**		*29*
tai80b	80	818415043 [a]	0.29	9/0	0.33	6/0	0.35	8/2	0.95	6/0	0.03	10/7	**0**		*138*
tai100b	100	1185996137 [a]	0.19	7/0	0.11	7/0	0.06	9/3	0.70	5/0	0.01	10/3	**0**		*430*
tai150b	150	498896643 [b]	0.39	9/0	0.54	7/0	0.40	8/0	0.55	6/0	0.11	10/2	**0.10**	10/2	*2300*

[a] comes from [3]; [b] comes from [27].

It can be seen that the quality of solutions depends on the type of problems being solved. For the random instances, the results are inferior to those for the real-life like instances; this indicates that these instances are much more hard to solve and still remain the great challenge for the researchers. Regarding the real-life like instances, they are relatively easy for many heuristics, among them, the hybrid genetic algorithms. Our extended hybrid genetic algorithm was able to find the best known (pseudo-optimal) solutions for all these instances (except the largest one) surprisingly quickly. For example, the average time needed to find the pseudo-optimal solution for the instance tai100b is equal to 400 seconds on 900 MHz computer. The results obtained show very promising efficiency of the proposed extension of HGA. In many cases, EHGA appears to be considerably superior to other efficient algorithms for both random and real-life like instances.

The results of EHGA may be improved even more by an accurate tuning of the control parameters. Of course, we can obtain higher quality solutions by increasing the total number of generations (N_{gen}), but at the cost of longer computation time. After an additional long-lasting experimentation, EHGA was successful in discovering new record-breaking solutions for two large random instances, namely, tai80a and tai100a. The new values of the objective function, which are better than those reported in [20], are equal to **13535624** and **21102912**, respectively.

4. Concluding remarks

In this paper, an extended hybrid genetic algorithm (EHGA) for the quadratic assignment problem is presented. The results obtained by EHGA demonstrate the excellent performance of the proposed algorithm with respect to the performance measures used. The main features of this algorithm are as follows: a) it incorporates an efficient tabu search algorithm as a local improvement procedure; b) the large population of solutions is not necessary: its compactness is fully compensated by the outstanding performance of TS; c) mutation operator is not needed in the GA itself, because the solutions undergo transformations in the TS procedure; d) a special restart mechanism implemented helps to overcome the loss of diversity within the population and the premature convergence of GA.

The idea of hybridization should further be exploited. The following directions for the possible improvement of EHGA may be proposed: 1) using the reactive tabu search instead of the straightforward (robust) tabu search (as a possibly more efficient local improvement procedure); 2) implementing other, more elaborated mutation (perturbation) operators within extended TS; 3) designing innovative crossover operators, for example, so-called multiple parent crossovers; 4) trying other restart strategies; 5) maintaining a mixture ("pout-pourri") of different local improvement procedures (like the descent local search, simulated annealing, tabu search, etc.) to allow flexible tuning (or, even self-tuning) of the genetic algorithm to the specific problem. Putting these directions into efficient implementations could be a 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] **A. Bölte, U.W. Thonemann.** Optimizing simulated annealing schedules with genetic programming. *European Journal of Operational Research*, 1996, *Vol.*92, 402–416.

[3] **R.E. Burkard, S. Karisch, F. Rendl.** QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 1997, *Vol.*10, 391–403.

[4] **V.D. Cung, T. Mautor, P. Michelon, A. Tavares.** A scatter search based approach for the quadratic assignment problem. *Proceedings of the IEEE International Conference on Evolutionary Computation and Evolutionary Programming* (*ICEC'97*), *Indianapolis, USA*, 1997, 165–170.

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

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

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

[8] **L.M. Gambardella, E. Taillard, M. Dorigo.** Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 1999, *Vol.*50, 167–176.

[9] **F. Glover.** Tabu search: part I. *ORSA Journal on Computing*, 1989, *Vol.*1, 190–206.

[10] **F. Glover.** Tabu search: part II. *ORSA Journal on Computing*, 1990, *Vol.*2, 4–32.

[11] **F. Glover, M. Laguna.** Tabu Search. *Kluwer, Dordrecht*, 1997.

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

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

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

[15] **Y. Li, P.M. Pardalos, M.G.C. Resende.** A greedy randomized adaptive search procedure for the quadratic assignment problem. *P.M.Pardalos, H.Wolkowicz (eds.), Quadratic Assignment and Related Problems*. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, *Vol.*16, *AMS*, *Providence*, 1994, 237–261.

[16] **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.

[17] **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.

[18] **Z. Michalewicz, D.B. Fogel.** How to Solve It: Modern Heuristics. *Springer, Berlin-Heidelberg*, 2000.

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

[20] **A. Misevicius.** A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, *in press*.

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

[22] **H. Mühlenbein.** Genetic algorithms. *E.H.L.Aarts, J.K. Lenstra (eds.), Local Search in Combinatorial Optimization, Wiley, Chichester*, 1997, 137–171.

[23] **S. Sahni, T. Gonzalez.** P-complete approximation problems. *Journal of ACM*, 1976, *Vol.*23, 555–565.

[24] **T. Stützle.** Iterated local search for the quadratic assignment problem. *Tech. Report AIDA*-99-03, *Darmstadt University of Technology, Germany*, 1999.

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

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

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

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