A Fast Hybrid Genetic Algorithm for the Quadratic Assignment Problem

Alfonsas Misevicius
Kaunas University of Technology
Studentu st. 50-400a/416a
LT-51368 Kaunas, Lithuania
Tel. 370-37-300372
alfonsas.misevicius@ktu.lt

ABSTRACT

Genetic algorithms (GAs) have recently become very popular by solving combinatorial optimization problems. In this paper, we propose an extension of the hybrid genetic algorithm for the well-known combinatorial optimization problem, the quadratic assignment problem (QAP). This extension is based on the "fast hybrid genetic algorithm" concept. An enhanced tabu search is used in the role of the fast local improvement of solutions, whereas a robust reconstruction (mutation) strategy is responsible for maintaining a high degree of the diversity within the population. We tested our algorithm on the instances from the QAP instance library QAPLIB. The results demonstrate promising performance of the proposed algorithm.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods.*

General Terms

Algorithms.

Keywords

Heuristics, genetic algorithms, tabu search, combinatorial optimization, quadratic assignment problem.

1. INTRODUCTION

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

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

One of the interpretations of the QAP is that of Koopmans and Beckmann [11]. In this case, one deals with locating n facilities on n locations (sites) with some physical products flowing between the facilities, and with distances between the locations. The element a_{ij} is the flow (cost) from the facility i to facility j, and 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). Thus, to solve the QAP one has to find an assignment that minimizes the "transportation cost" between facilities, z.

It has been proved that the QAP is NP-hard [20]. Therefore, heuristic algorithms are to be used for solving medium- and large-scale QAPs in reasonable time. Various methods such as ant algorithms [22], greedy randomized adaptive search procedures [12], iterated local search [14], scatter search [4], simulated annealing [1], tabu search [17,21] have been tried. Starting from 1994, several researchers have applied the genetic algorithms (GAs) to the QAP, first of all [6,7,13,15,16,24].

The remaining part of this paper is organized as follows. In Section 2, the state-of-the-art and further extensions of genetic algorithms are outlined. The new proposed hybrid genetic algorithm for the quadratic assignment problem is discussed in Section 3. In Section 4, we present computational results obtained by examining a number of instances of the QAP. Finally, Section 5 completes the paper with concluding remarks.

2. GENETIC ALGORITHMS

The basic definitions related to combinatorial optimization (CO) problems and genetic algorithms are as follows. Let S be a set of feasible solutions (solution space) of a combinatorial optimization problem with a real-valued objective function f: $S \to \Re$. Without loss of generality, we assume that f seeks a global minimum. Solving the CO problem means searching for a solution $s_{\text{opt}} \in S$ such that

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

The solution s_{opt} is called a globally optimal solution (the set $S_{\text{opt}} \subseteq S$ denotes the set of global optima). In addition, a neighbourhood function Θ: $S \to 2^S$ is defined. It attaches for every $s \in S$ a set of neighbouring solutions $\Theta(s) \subseteq S$. Each solution $s' \in \Theta(s)$ can be reached from s by an operation called a move. The 2-exchange neighbourhood function Θ_2 is widely used for the permutation-based solutions. For a given 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'. One of the ways of defining the distance for the QAP solutions is counting the items that are assigned to different positions of the solutions, i.e.

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

2.1 Standard Genetic Algorithms

The original concepts of the genetic algorithms were developed by Holland [10] as far back as 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 ($f(s_i)$). The less the objective function value, the larger 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 pair of members 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, and mutation procedures. The detailed material on this and related topics can be found in [5,9,19].

2.2 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 constructive 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. These additional components, plus the standard genetic algorithm are what one calls a hybrid genetic (memetic) algorithm (HGA) [18].

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 algorithm. The result of this step is a population which represents a collection of locally optimal solutions. After the creation of the 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 it operates with the optimized solutions as the inputs to the crossover operator. Although the crossover operator is highly responsible for the efficiency of a genetic algorithm, 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 (i.e. local improvement) procedure to be applied to each offspring to obtain again locally optimal solutions.

The mutation is then typically performed on the locally optimized offspring. It should be stressed that 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.

The hybridization of GA is an essential improvement over the standard GA. This is mainly due to the fact that 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 than when searching in a random space. For more details of HGAs, the reader is addressed to [18].

2.3 Fast Hybrid Genetic Algorithms

Applying the hybrid genetic algorithms still does not necessary mean that high quality solutions are reached at reasonable time. Indeed, HGAs often use the elaborated improvement heuristics (like simulated annealing, tabu search, etc.) that in general are quite time-consuming. This could be thought of as a serious shortcoming, especially if we wish to create HGAs that are competitive with other intelligent optimization techniques. In these circumstances, it is important to further exploit the idea of hybridization, which might result in designing new improved HGAs. A concept entitled "*fast hybrid genetic algorithms*" (FHGAs) may be proposed as one of the possible extensions of HGAs. The following are the basic preliminary principles (features) that should guide the design of FHGAs.

I. FHGAs should incorporate fast (robust) local improvement algorithms (heuristics). Here, we assume that the algorithm A_1 is "faster" than the algorithm A_2 if A_1 finds (in average) the solution(s) with the average objective function value (quality) $\bar{f}$ in less time (t_1) than A_2. Note that this definition does not mean that the best (not averaged) solution found by A_1 during the time interval t_1 is necessary better than the best one found by A_2. Moreover, A_1 might not be necessary better than A_2 at longer runs. Naturally, the long time behaviour does not matter as long as we are speaking of the fast algorithms in the context of FHGAs. The only matter is how quickly an algorithm achieves the solution(s) with the given quality $\bar{f}$. (The value of $\bar{f}$ may be determined experimentally. It should be neither too large, nor too small. In the first case, it might be complicated to decide which algorithm is better; in the second case, there is a danger that $\bar{f}$ will never be achieved.)

II. In FHGAs, the compactness of the population is of great importance. As long as powerful local improvement procedures are used, the large populations of solutions are not necessary at all: the small size of the population is fully compensated by the robustness of the heuristic improvement algorithm. Obviously, the compact populations allow to save the computation time when comparing to other HGAs, which deal with larger populations. However, note that if the size of the population is very small, FHGA may evolve to an ordinary single-solution like algorithm. The preliminary results of the experimentation on the QAP show that this size is proportional to $\sqrt{n}$, where n is the problem size.

III. FHGAs must maintain a high degree of the diversity within the population. The loss of the diversity (which in turn results in a "stagnation" and premature convergence) is the thing to avoid also in the classical HGAs. In the environments of the robust improvement heuristics, the problem of preserving the individuals' diversity is even more severe. At first look, there is a conflict. How the principle (III) can be guaranteed within small populations? According to naive thinking, the smaller the size of the population, the larger the probability that the diversity will be lost quickly by using the efficient improvement heuristics. To fight this difficulty, the strategy based on a reconstruct and improve (R&I) principle may be helpful. The primary goal of R&I was to improve the performance of the single-solution based heuristics, like descent local search or simulated annealing. There are two main phases in the R&I paradigm: 1) the reconstruction phase (it can also be viewed as diversification of the search), and 2) the local improvement phase (it can be viewed as intensification of the search). At the first phase, the existing solution is reconstructed (mutated) in an appropriate way. In the second phase, one tries to improve the solution just reconstructed as best as one can; hopefully, the new improved solution is better than the solutions obtained in the previous iterations. By repeating these phases many times one seeks for high quality results.

We can use the similar policy for the population-based algorithms. It is enough to think of the populations as some kind of meta-solutions. So, instead of the reconstruction/improvement of the single solutions, the meta-solutions, i.e. the whole populations are to be reconstructed/improved. Basically, this means that all (or the majority of) the members of the population undergo reconstruction/improvement. This strategy may also be viewed as a special case of island-based genetic algorithms [3], which deal with several "parallel" sub-populations. In R&I approach, however, the populations are processed rather in a "sequential" way. That is, at every generation, the test is performed, whether the level of the diversity of the current population is below a certain threshold. If it is, then a new population is created by reconstruction of the actual population (or even by restart from a randomly generated population); otherwise, the search is continued with the current population. The resulting paradigm of FHGA is presented in Figure 1.

3. A FAST HYBRID GENETIC ALGORITHM FOR THE QAP

3.1 Initial Population Generation

The algorithm starts with the creation of an initial population P. This is done in two steps: firstly, $PS=|P|$ solutions (permutations) are generated in a pure random way; secondly, all the individuals

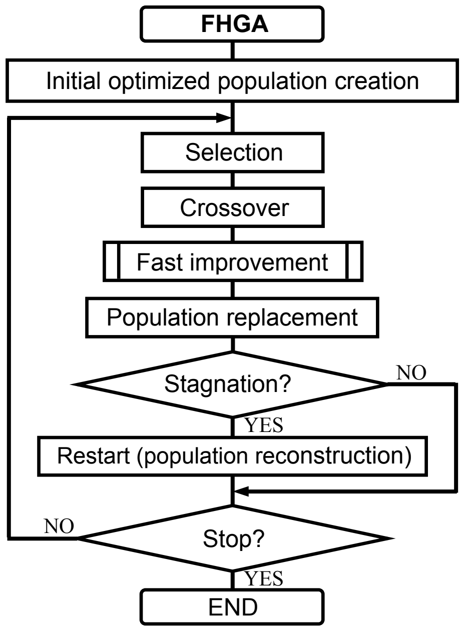

Figure 1. Basic flowchart of FHGA.

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. An enhanced tabu search (TS) procedure (see below) is used in the role of the fast local improvement. So, the input of the further genetic process is a population that consists of PS locally optimal solutions.

3.2 Parents Selection Rule

After creating the initial population, the algorithm proceeds in the following way. Two solutions are selected to be parents of a new individual (child). For the selection, we apply a rank based rule [24]. The position, u, of the parent within the sorted population is determined by 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.

3.3 Crossover Operator

For the parents merging, we use a variant of the crossover operator proposed in [24] (we call it 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 ith position of the parents and child, respectively). Second, the unassigned positions of the child 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 solution). The ULX operator implies a high degree of

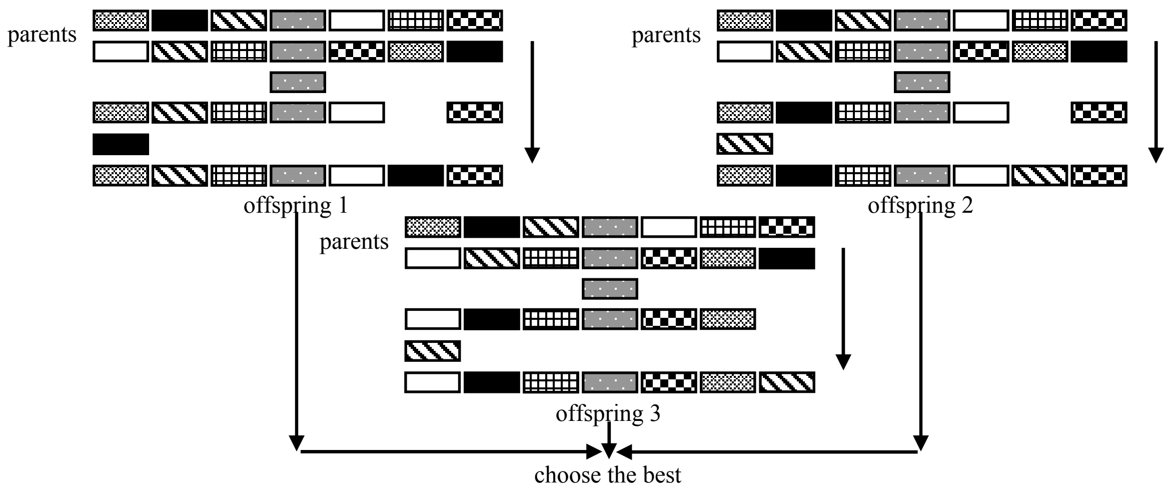

Figure 2. Example of producing a child in the "triplet" crossover.

randomness. Even the same pair of parents may produce lots of quite different children, especially, when the distance between the parents is large. So, we can extend the functioning of the crossover by making it create "*m*-plets" (*m*>1) 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 2). We call this type of proceeding an elite crossover (EX).

The search could 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).

3.4 Improvement Procedure

As mentioned in Sections 2.2, 2.3, genetic algorithms succeed in search if only they dispose of a robust local improvement 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 [8] (this is especially true for the quadratic assignment problem [17,21]). Consequently, we have chosen this approach as a perfect compromise between the opposites mentioned. In particular, we use the enhanced tabu search (ETS) algorithm proposed by Misevicius [17].

The main idea behind ETS is that the classical TS (it serves as an intensification mechanism) is combined with the appropriate perturbations of solutions (they serve as a diversification mechanism). In particular, the modified robust tabu search algorithm due to Taillard [21] is in the role of intensification. Very roughly, this algorithm can be outlined in the following way. Initialize the tabu list $\boldsymbol{T}$, and start from the current solution π. Then, continue the following process until a predefined number of steps, τ, have been performed:

— find a neighbour π'' of the current solution π in such a way that $\pi'' = \arg\min_{\pi' \in \Theta_2^{\Diamond}(\pi)} z(\pi')$,

where $\Theta_2^{\Diamond}(\pi) = \{\hat{\pi} \mid \hat{\pi} \in \Theta_2(\pi), \hat{\pi} = \pi \oplus m_{ij} \text{ and} ((m_{ij} \text{ is not tabu}) \text{ or}$

$$(f(\hat{\pi}) < f(\pi^*)))\},$$

where π^* is the best so far solution (we utilize the compact notation m_{ij} ($i, j = 1, 2, ..., n$) for the move from π to $\pi' \in \Theta_2(\pi)$, which exchanges exactly ith and jth elements in the permutation π ($\pi' = \pi \oplus m_{ij}$ means that π' is obtained from π by applying m_{ij}));

— update the tabu list $\boldsymbol{T}$ by including the move m_{uv}, where m_{uv} is the move from the solution π to the solution π'';

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

We propose to add an additional component to the above procedure. Our idea is to embed an alternative intensification mechanism based on a pure steepest descent (SD) algorithm. So, if the inequality $z(\pi'') - z(\pi) < 0$ holds, the basic intensification procedure is temporally "interrupted" in order to apply the SD procedure. However, the alternative intensification procedure is omitted if it already took place within the last k steps. The parameter k can be related to the problem size n: $k = \omega n$, where ω is the alternative intensification frequency factor (in our algorithm, we use $\omega = 0.5$). Note that all the moves performed during the SD procedure are stored in the tabu list.

The results obtained demonstrate that this combined tabu search and steepest descent (TS-SD) strategy performs well in both "standalone" and "genetic algorithm component" mode. In addition, we usually apply rather short runs of TS-SD. We refer this strategy to as a limited tabu search (LTS). Firstly, LTS allows saving the computation time, which is crucial for FHGA. On the other hand, LTS in combination with the other genetic operators is quite enough to seek for high quality solutions.

Regarding the diversification mechanism, we use the random pairwise interchange mutations (rpi-mutations). They can simply be seen as sequences of moves $m_{r_1 r_2}, m_{r_3 r_4}, ..., m_{r_{2\mu-1} r_{2\mu}}$. All one

```
procedure FastHybridGeneticAlgorithm;
  // input: $A,B$ – the flow and distance matrices, $n$ – the problem size; output: $\pi^*$ – the best solution (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, $\xi_1$, $\xi_2$ – the mutation factors //
  $\mu_{min} := \max(2, \lfloor \xi_1 n \rfloor)$; $\mu_{max} := \max(2, \lfloor \xi_2 n \rfloor)$;
  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 enhanced tabu search procedure EnhancedTabuSearch;
  $\pi^* := \operatorname{argmin}_{\pi \in P} z(\pi)$ ; // $\pi^*$ is the best so far solution //
  for $i$ := 1 to $N_{gen}$ do begin // main cycle //
    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 elite crossover EX to $\pi'$ and $\pi''$, get the offspring $\dot{\pi}$ ;
      $\pi^\bullet$ := EnhancedTabuSearch($\dot{\pi}$); // the offspring is improved by using $Q$ iterations of ETS //
      add the improved solution $\pi^\bullet$ 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 FHGA) //
    end; // for $j$ //
    cull the population $P$ by removing $N_{cross}$ worst individuals;
    if the diversity of $P$ is below predefined threshold then make "restart" in two phases:
        (i) reconstruct (mutate) all the members of $P$, except the best one,
       (ii) improve each reconstructed solution by using EnhancedTabuSearch
            (save the new best encountered solution if any)
  end // for $i$ //
end.
```

Figure 3. Template of the fast hybrid genetic algorithm for the QAP.

needs by rpi-mutation is to generate the pairs of random numbers (r_i, r_{i+1}), such that $1 \le r_i, r_{i+1} \le n$, $r_i \ne _{i+1}$. The larger the length of the sequence (i.e. a mutation level) μ, the stronger the mutation, and vice versa. In turn, the stronger the mutation, the larger the distance between the current solution and the mutated one. One can add more robustness to the diversification process by performing "concentric" mutations. In this case, the mutation level μ changes 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. (The values $\mu_{\min}$, $\mu_{\max}$ can be related to the problem size n, i.e. $\mu_{\min} = \max\{2, \lfloor \xi_1 n \rfloor\}$ and $\mu_{\max} = \max\{2, \lfloor \xi_2 n \rfloor\}$, where ξ_1, ξ_2 $(0 < \xi_1 \le \xi_2 \le 0.5)$ are the user-defined coefficients (mutation factors). 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 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) is accepted for the mutation, no matter its quality. The advantage of this strategy is allowing to search in many distinct regions of the solution space.

The combination of intensification and diversification is done according to $(Q,\tau,1)$-scheme. Within 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-SD and one call to the mutation procedure are performed. In this case, the aspect ratio Q/τ may be seen as a measure for the mutation frequency. The value of Q/τ is quite important. Suppose, $Q\tau = \text{const}$. Then, the larger the value of Q/τ, the larger the mutation frequency, and vice versa. The optimal value of Q/τ can 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 example, for random QAP data (see Section 4), we found that the optimal value is somewhere between 0.05 and 5.

Finally, note that the ETS procedure is applied in three situations: (a) the initial population creation; (b) the inner cycle of FHGA; (c) the restart (see Section 3.5). The numbers of the ETS and TS-SD iterations are different for these cases. We operated with the following numbers: $Q_{\text{ini}} = Q$, $\tau_{\text{ini}} = \frac{1}{2} n\tau$, $Q_{\text{restart}} = Q$, $\tau_{\text{restart}} = \frac{1}{2} n\tau$ (for a random data set); $Q_{\text{ini}} = 4Q$, $\tau_{\text{ini}} = \tau$, $Q_{\text{restart}} = 3Q$, $\tau_{\text{restart}} = \tau$ (for a real world data set) (here, Q_{ini}, τ_{ini}, Q, τ, Q_{restart}, τ_{restart} are the numbers of ETS and TS-SD iterations in the situations (a), (b), and (c), respectively).

3.5 Population Replacement Scheme

The 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, FHGA proceeds with the new population as described above. An extra dealing with the

```
function EhancedTabuSearch($\pi$);
  // input: $\pi$ – the current solution (permutation); output: $\pi^*$ – the best solution found; parameters: $Q$, $\tau$, $\mu_{min}$, $\mu_{max}$ //
  apply $\tau$ iterations of the robust tabu search to $\pi$, get the improved solution $\pi^{\bullet}$;
  $\pi := \pi^{\bullet}$; $\pi^* := \pi^{\bullet}$; $\mu := \mu_{min} - 1$;
  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 solution $\tilde{\pi}$;
    apply $\tau$ iterations of the combined robust tabu search and steepest descent to $\tilde{\pi}$,
    get the new improved solution $\pi^{\bullet}$;
    if $z(\pi^{\bullet}) < z(\pi^*)$ then begin
      reset the mutation level $\mu$; $\pi^* := \pi^{\bullet}$ // save the best so far solution (as a possible result of ETS) //
    end
  end; // for //
  return $\pi^*$
end.
```

Figure 4. Template of the enhanced tabu search for the QAP.

population (i.e. a restart mechanism) is applied under some circumstances. In particular, 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. In fact, there are two phases at the restart: a) the reconstruction (mutation) of the members of the population; b) the local improvement of the reconstructed solutions by means of ETS. For the reconstruction, we tried the following deep mutation procedure. At the first step, we apply the mutation operator very similar to that used in the ETS algorithm. The main difference lies in the increased mutation level; more precisely, the mutation is done in such a way that $\rho(\pi, \tilde{\pi}) = n$, where $\pi, \tilde{\pi}$ are the solutions before and after mutation; n is the problem size. The solution $\tilde{\pi}$ then undergoes a so-called shift mutation (s-mutation) at the second step. The reason of this additional mutation is to achieve even more diversity. The s-mutation is very simple to implement: having a permutation, just shift all the items to right (in a wrap-around fashion) by a predefined number of positions, θ $(1 \le \theta < n)$. The value of θ may be varied in a cyclic way, i.e. $\theta_i = (\theta_{i-1} \bmod (n-1)) + 1$, where i denotes the current mutation call number, and $\theta_0 = 1$. This guaranties that, for every member of the population, the different value of θ is used as long as the population size, PS, is less than the problem size, n.

After restart, FHGA goes on in an ordinary way. The overall process is continued in an iterative way until a given number of generations (N_{gen}) have been performed. The generalized templates of the fast hybrid genetic algorithm, as well as the enhanced tabu search procedure are presented in Figures 3 and 4.

4. 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 publicly available library QAPLIB [2]. Pentium 900 MHz computer was used in the experiments. 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 [21]; 2) fast ant system (FANT) [22]; 3) genetic hybrid (GH) algorithm [7]; 4) genetic algorithm due to Lim, Yuan and Omatu (GA-LYO) [13]; 5) improved hybrid genetic algorithm (IHGA) [16]. 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 runs (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 runs) – $C_{1\%}$; c) the number of the best known solutions found – C_{bks}.

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 FHGA are collected in Table 1. Note that 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) for

Table 1. Values of the parameters for FHGA

Problem type	PS	N_{gen}	σ	N_{cross}	Q	τ	ξ_1	ξ_2
(a)	$\lfloor\sqrt{n}\rfloor$	$\frac{1}{4}n$	1.3	$\frac{1}{4}PS$	5	$\frac{1}{2}n^2$	depends on problem[a]	depends on problem[b]
(b)	$\lfloor\sqrt{n}\rfloor$	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

Instance	n	BKV	$\bar{\delta}$, $C_{1\%}/C_{\mathrm{bks}}$						CPU time (sec.)
			RTS	FANT	GH	GA-LYO	IHGA	FHGA	
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**	*30*
tai40a	40	3139370 [a]	0.46 8/0	1.55 2/0	0.62 9/0	1.95 0/0	0.23 10/**1**	**0.21** 10/**1**	*68*
tai50a	50	4941410 [a]	0.79 5/0	1.76 1/0	0.87 4/0	2.01 0/0	0.42 10/1	**0.32** 10/**2**	*330*
tai60a	60	7205962 [b]	0.84 2/0	1.71 0/0	1.01 2/0	1.93 0/0	0.55 9/0	**0.35** 10/**1**	*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	**0.26** 10/0	*3600*
tai100a	100	21123042 [b]	0.59 8/0	1.19 6/0	0.51 8/0	1.16 2/0	0.31 10/0	**0.19** 10/0	*12000*

[a] comes from [2]; [b] comes from [17].

the intensification process; whereas, for the real-life like instances, intensification is relatively less important than diversification (see also [17]). 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. 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 the random instances are much more hard to solve and still remain a great challenge for the researchers. Regarding the real-life like instances, they are relatively easy for many heuristics, among them, the genetic algorithms. Our fast genetic algorithm was able to find the 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 merely 400 seconds on Pentium 900 MHz computer. In general, FHGA appears to be superior to other efficient algorithms for both random and real-life like instances and should be considered as a promising extension of HGA.

The results of FHGA may be improved even more by an accurate tuning of the control parameters. Of course, we can obtain better solutions by increasing the total number of generations N_{gen}, but at the cost of longer computation time. After additional long-lasting experimentation, FHGA was successful in discovering new record-breaking solutions for two large random QAP instances tai80a and tai100a. The new values of the objective function, which are better than those reported in [17], are equal to **13527962** and **21075606**, respectively. The new best known solution was also found for one of the instances of the grey pattern problem (GPP), which is a special case of the QAP. The graphical illustration of the new solution is shown in Figure 5.

5. CONCLUDING REMARKS

In this paper, a variant of the fats hybrid genetic algorithm (FHGA) for the quadratic assignment problem is presented. The results obtained by FHGA 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 the enhanced tabu search (ETS) 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 ETS; c) mutation operator is not needed in the GA itself, because the solutions undergo transformations in the ETS procedure; d) a restart mechanism

Table 3. Comparison of the algorithms on real-life like instances

Instance	n	BKV	$\bar{\delta}$, $C_{1\%}/C_{\mathrm{bks}}$						CPU time (sec.)
			RTS	FANT	GH	GA-LYO	IHGA	FHGA	
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**	*4.8*
tai50b	50	458821517 [a]	0.24 10/0	0.22 9/0	0.03 10/7	0.95 5/0	0.01 10/8	**0**	*18*
tai60b	60	608215054 [a]	0.30 10/0	0.18 9/3	0.02 10/6	0.80 4/0	0.02 10/9	**0**	*28*
tai80b	80	818415043 [a]	0.29 9/0	0.33 6/0	0.35 8/2	0.95 6/0	0.02 10/7	**0**	*136*
tai100b	100	1185996137 [a]	0.19 7/0	0.11 7/0	0.06 9/3	0.70 5/0	0.03 10/5	**0**	*400*
tai150b	150	498896643 [b]	0.39 9/0	0.54 7/0	0.40 8/0	0.55 7/0	0.11 10/2	**0.05** 10/**3**	*2000*

[a] comes from [2]; [b] comes from [23].

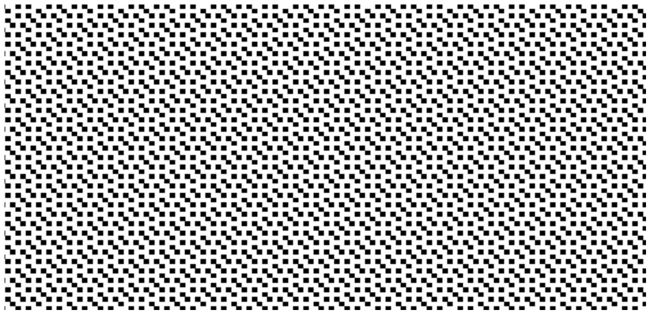
Figure 5. Pseudo-optimal grey frame of density 72/256.

helps avoiding the loss of the diversity within the population and the premature convergence of GA.

The idea of fast hybrid GAs should further be exploited. The following directions for the improvement of FHGA may be interesting: 1) using the reactive tabu search instead of the enhanced tabu search (as a possibly more efficient local improvement procedure); 2) implementing other more elaborated mutation operators within ETS; 3) designing innovative crossover operators, for example, multiple parent crossovers; 4) trying other restart policies; 5) maintaining a mixture of different local improvement heuristics to allow the flexible tuning of the genetic algorithm to the particular problem.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] Bölte, A., and Thonemann, U. W. Optimizing simulated annealing schedules with genetic programming. *European Journal of Operational Research*, *92*, 1996, 402-416.

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

[3] Calègari, P., Coray, G., Hertz, A., Kobler, D., and Kuonen, P. A taxonomy of evolutionary algorithms in combinatorial optimization. *Journal of Heuristics*, *5*, 1999, 145-158.

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

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

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

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

[8] Glover, F., and Laguna, M. *Tabu Search*. Kluwer, Dordrecht, Netherlands, 1997.

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

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

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

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

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

[14] Lourenco, H. R., Martin, O., and Stützle, T. Iterated local search. In *Handbook of Metaheuristics*, F.Glover and G.Kochenberger, Eds. Kluwer, Norwell, MA, 2002, 321-353.

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

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

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

[18] Moscato, P. Memetic algorithms: a short introduction. In *New Ideas in Optimization*, D.Corne, M.Dorigo, and F.Glover, Eds. McGraw-Hill, London, UK, 1999, 219-234.

[19] Mühlenbein, H. Genetic algorithms. In *Local Search in Combinatorial Optimization*, E.H.L.Aarts and J.K.Lenstra, Eds. Wiley, Chichester, UK, 1997, 137-171.

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

[21] Taillard, E. Robust taboo search for the QAP. *Parallel Computing*, *17*, 1991, 443-455.

[22] Taillard, E. FANT: fast ant system. Technical Report IDSIA-46-98, Lugano, Switzerland, 1998.

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

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