Available online at www.sciencedirect.com
SCIENCE DIRECT®
Knowledge-Based Systems 17 (2004) 65–73
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

An improved hybrid genetic algorithm: new results for the quadratic assignment problem

Alfonsas Misevicius

Department of Practical Informatics, Kaunas University of Technology, Studentu St. 50-400a, LT-3031 Kaunas, Lithuania

Available online 31 March 2004

Abstract

In this paper, we propose an improved hybrid genetic algorithm (IHGA). It uses a robust local improvement procedure as well as an effective restart mechanism that is based on so-called 'shift mutations'. IHGA has been applied to the well-known combinatorial optimization problem, the quadratic assignment problem (QAP). The results obtained from the experiments on different QAP instances show that the proposed algorithm appears to be superior to other approaches that are among the best algorithms for the QAP. The high efficiency of our algorithm is also corroborated by the fact that new record-breaking solutions were obtained for a number of large real-life instances.

Keywords: Genetic algorithms; Combinatorial optimization; Quadratic assignment problem

1. Introduction

Genetic algorithms (GAs) are population based heuristic approaches. They have been applied successfully in various domains of artificial intelligence, search, and optimization. The promising results were obtained for many difficult optimization problems, for example, continuous optimization [1], graph partitioning [2], network design problem [3], scheduling problems [4,5], set covering problem [6], traveling salesman problem [7]. (GAs for the quadratic assignment problem (QAP) are discussed in Section 3.)

The principles of GAs were developed by Holland [8]. Very briefly, GAs can be characterized by the following features: (a) a mechanism for selecting individuals (corresponding to solutions of the optimization problem) from the population; (b) an operator for creating new individuals, i.e. offsprings by combining the information contained in the previous individuals; (c) a procedure for generating new solution by random perturbations of the single previous solutions; and (d) a rule for updating the population (culling solutions from the current population). These features are referred to as selection, crossover (recombination), mutation, and culling (updating), respectively.

There exists a great variety in the choice of how select, cross, mutate, and cull the individuals (for various modifications, see, for example, Refs. [9–11]). Regarding QAP, the detailed implementations of selection, crossover, mutation, and culling procedures are described in Section 3.2.

The remaining part of this paper is organized as follows. In Section 2, basic ideas of a hybrid genetic approach are outlined. Section 3 describes the improved hybrid genetic algorithm (IHGA) to the QAP. Also, the results of computational experiments are presented. Section 4 completes the paper with concluding remarks.

2. An improved hybrid genetic algorithm: basic concepts

In contrast to the classical GAs that rely on the concept of biological evolution, the modern GAs, known as hybrid GAs (or memetic algorithms), are based rather on ideas evolution [12]. In the context of optimization, a GA is called hybrid if solutions of a given problem are improved by a local optimization (local search) algorithm—like the ideas are modified by the person before passing to the next generation. This means that populations consist solely of locally optimal solutions.

The hybrid GA succeeds in search only if it disposes of an efficient local optimization procedure. We propose to use an iterated local search (ILS) technique [13] (very similar to a variable neighbourhood search [14] and ruin and recreate (R&R) approach [15]). The central idea of the ILS method is to obtain high quality results by reconstruction (destruction) of an existing solution and a following improving

Tel.: +370-37-30-03-72; fax: +370-37-30-03-52.
E-mail address: alfonsas.misevicius@ktu.lt (A. Misevicius).

0950-7051/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2004.03.001

procedure, i.e. ILS can be thought of as an iterative process of reconstructions and improvements applied to solutions. This technique appears to be superior to the well-known random multistart method in most cases. The advantage is that, instead of generating new solutions, a better idea is to reconstruct (a part of) the current solution: for the same computation time, many more reconstruction improvements can be done than when starting from purely random solutions, because the reconstruction improvement requires only a few steps to reach the next local optimum; a new local optimum can be found very quickly-usually faster than when starting from a randomly generated solution.

The favourable feature of the iterated search is that it includes reconstructions that can be viewed as mutations. This helps considerably when incorporating ILS into the hybrid GA: there is no need in mutation operator within GA itself (excepting only the case discussed below) since each individual already undergoes some transformations in the ILS procedure. Another important aspect is related to the size of population. As long as the ILS procedure is distinguished for the outstanding performance, the large populations of solutions are not necessary at all: the small size of the population is fully compensated by the ILS procedure. So, this feature allows to save the computation (CPU) time when comparing to other population-based algorithms.

It should also be noted that, in the hybrid GA proposed, a so-called restart mechanism [16] is used; this means that, if the fact of a premature convergence of the algorithm is determined, then a specific procedure ('cold restart') takes place. In fact, this restart consists of a special type mutations (referred to as 'shift mutations') to be applied to all the individuals but the best. (The shift mutation procedure is described in Section 3.2.)

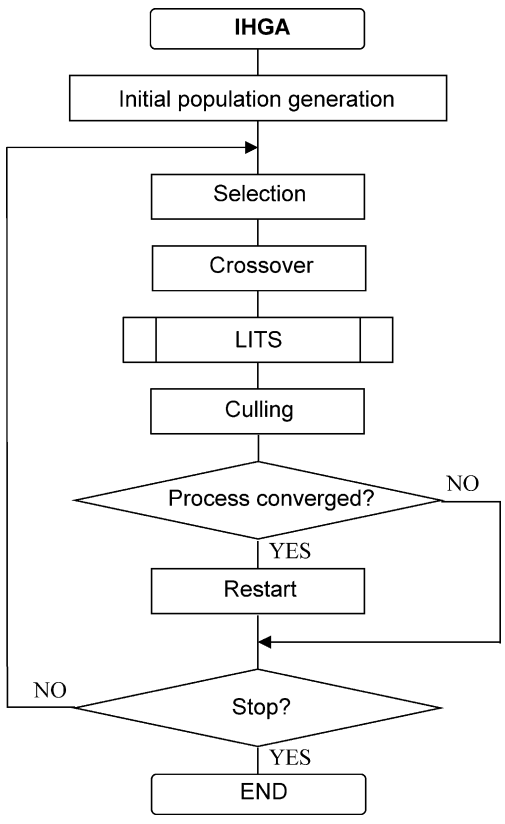


Fig. 1. Basic flowchart of the improved hybrid genetic algorithm.

Returning to improvement of solutions within ILS, the different improving procedures can be used, for example, hill climbing, simulated annealing (SA) or tabu search (TS). The last one seems to be the best candidate for the role of the local improvement: the power of TS has been demonstrated for a variety of optimization problems [17]. This is due to the robustness, efficacy and quickness of the TS procedure. (The basic principles of TS will briefly be considered in Section 3.2.) Note that, as a rule, we use exclusively short runs of the TS—we call this approach a limited iterated tabu search (LITS). Many simulations have demonstrated that these limited iterations allow saving the CPU time; on the other hand, LITS in combination with other genetic operators is quite enough to seek for near-optimal solutions.

The basic flowchart of the resulting hybrid GA—IHGA—is presented in Fig. 1. More thoroughly, we discuss the components of this algorithm in Section 3 in the context of the QAP.

3. New results for the quadratic assignment problem

3.1. The quadratic assignment problem

The QAP is formulated as follows. Given two matrices $\mathbf{A} = (a_{ij})_{n\times n}$ and $\mathbf{B} = (b_{kl})_{n\times n}$ and the set Π of permutations of the integers from 1 to n, find a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n)) \in \Pi$ that minimizes

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

The important areas of the application of the QAP are: campus planning [18], computer-aided design (placement of electronic components) [19], image processing (design of grey patterns) [20], ranking of archaeological data [21], typewriter keyboard design [22], etc. [23]. For example, in the design of grey patterns (frames) [20], in order to get a grey frame of density m/n, one can generate a grid containing $n (n = n_1 \times n_2)$ square cases with m black cases and $n-m$ white cases. By juxtaposing many of these grids, one gets a grey surface of density m/n. To get the finest frame, the black cases have to be spread as regularly as possible. (Some examples of grey frames will be presented in Section 3.3.)

It has been proved that the QAP is NP-hard [24], therefore heuristic approaches have to be used for solving medium- and large-scale QAPs: ant algorithms [25], SA [26], TS [27]. Starting from 1994, several authors applied GAs to the QAP, first of all [28–34].

```
procedure improved_hybrid_genetic_algorithm
  generate initial population P⊂ Π // the population size is equal to PS, where PS≤n //
  P• := ∅ // P• – the population containing the improved (optimized) permutations //
  foreach π ∈ P do begin
    π• := lits(π) // lits – the function that performs the limited iterated tabu search on π//
    P• := P• ∪ {π•}
  end // foreach //
  π* := argmin z(π) // π* – the best permutation found //
        π∈P•
  repeat // main cycle of the genetic algorithm: generations cycle //
    sort the members of the population P• according the ascending order of the fitness
    for i := 1 to Q_cross do begin // cycle of offsprings generation and improvement //
      select π′,π″ ∈ P•
      apply optimized crossover to π′ and π″, get offspring π_offspr
      π• := lits(π_offspr) // improvement of the offspring //
      P• := P• ∪ {π•}
      if z(π•) < z(π*) then π* := π•
    end // for //
    remove k=Q_cross worst individuals from population P• // culling of the population //
    if population P• converged then begin
      P := P•, P• := ∅
      foreach π ∈ P \ bestof(P) do begin
        π~ := shift-mutation(π), P• := P• ∪ {lits(π~)}
      end // foreach //
    end // if //
  until maximum generation number is reached
end // improved_hybrid_genetic_algorithm //
```

Fig. 2. Template of the improved hybrid genetic algorithm for the QAP.

3.2. An improved hybrid genetic algorithm for the quadratic assignment problem

The template of the IHGA for the QAP (in an algorithmic language form) is shown in Fig. 2. The details are described below in this section.

The initial population $P(P \subset \Pi)$ is obtained by generating PS ($PS \leq n$) random permutations that are improved by the LITS procedure (see below).

For the parents selection, we apply a rank based selection rule [34]. This means that the better the individual, the larger probability of selecting it for the crossover.

The crossover operator used is based upon a uniform like crossover (ULX) [34]. It works as follows. First, all items assigned to the same position in both parents are copied to this position in the child. Second, the unassigned positions of a permutation are scanned from left to right: for the unassigned position, an item is chosen randomly, uniformly from those in the parents if they are not yet included in the child. Third, remaining items are assigned at random. In IHGA, we have slightly improved this classic scheme to get a so-called optimized crossover (OX). An optimized crossover is a crossover that (a) is ULX and (b) produces a child that has the smallest objective function among the children created by M runs of ULX. So, only the best (elite) child among M children is passed through the subsequent improvement procedure. Usually, the value of M is between $0.5n$ and $2n$ (n is the problem size). The number of

```
function lits(π) // limited_iterated_tabu_search //
  set τ to the number of steps of tabu search procedure
  apply tabu search procedure to π, get the resulting permutation π*, set π to π*
  for q := 1 to Q_lits do begin // main cycle consisting of Q_lits iterations //
    set μ to the current reconstruction (mutation) level
    apply reconstruction procedure to π, get the resulting permutation π~
    apply tabu search procedure to π~, get the resulting permutation π*
    if π* is better than π then π := π*
  end // for //
  return π
end // lits //
```

Fig. 3. Template of the limited iterated tabu search procedure for the QAP.

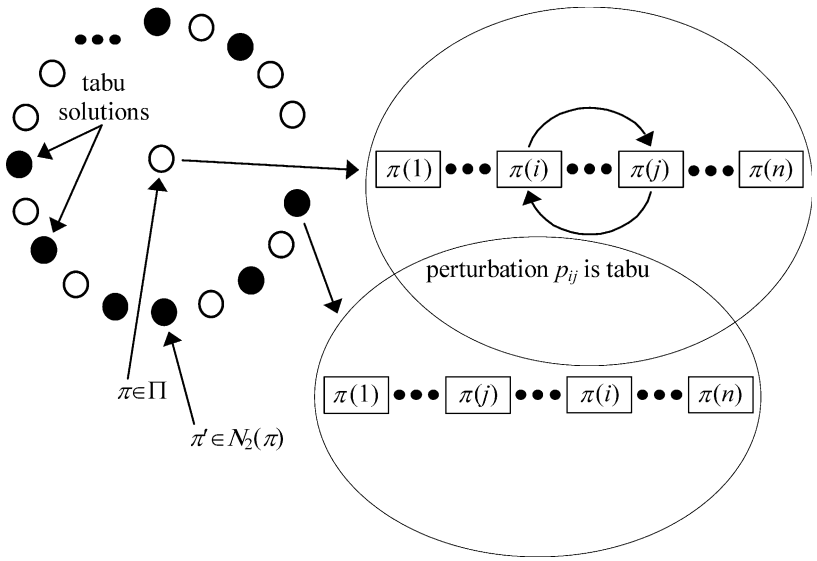


Fig. 4. Illustration of the two-exchange neighbourhood and tabu search.

optimized crossovers, i.e. elite offsprings per one generation can be controlled by a parameter, Q_{cross}. Typically, the value of Q_{cross} depends on the size of a population.

The limited iterated TS procedure (presented in Fig. 3) contains two main components: a TS based local search procedure, and a solution reconstruction procedure.

Before describing the TS procedure for the QAP, we give some very basic definitions. Let $N_2 : \Pi \rightarrow 2^{\Pi}$ be a two-exchange neighbourhood function that defines for each $\pi \in \Pi$ a set of neighbouring solutions of $\pi, N_2(\pi) = \{\pi' | \pi' \in \Pi, d(\pi, \pi') \leq 2\}$ ($d(\pi, \pi')$ is the 'distance' between solutions π and $\pi' : d(\pi, \pi') = \sum_{i=1}^{n} \text{sgn}|\pi(i) - \pi'(i)|$). A move (transformation) from the current solution π to the neighbouring one $\pi' \in N_2(\pi)$ can formally be defined by using a special operator, $p_{ij}(i, j = 1, 2, \ldots, n) : \Pi \rightarrow \Pi$, which exchanges ith and jth elements in the current permutation (Fig. 4). (Notation $\pi' = \pi \oplus p_{ij}$ means that π' is obtained from π by applying p_{ij}.) The TS algorithm can then be outlined as follows (for the detailed template, see Ref. [35]). Initialize tabu list, **T**, and start from an initial permutation π. Then, continue the following process until a predetermined number of steps, τ, have been performed:

(i) find a neighbour π'' of the current permutation π in such a way that

$$\pi'' = \arg\min_{\pi' \in N'_2(\pi)} z(\pi'),$$

where

$$N'_2(\pi) = \{\hat{\pi} | \hat{\pi} \in N_2(\pi), \hat{\pi} = \pi \oplus p_{ij} \text{ and } p_{ij} \text{ is not tabu}\};$$

(ii) replace the current permutation π by the neighbour π'', and use as a starting point for the next step;
(iii) update the tabu list **T**.

A reconstruction (a random pairwise interchanges based mutation) of the current solution can simply be achieved by performing some number, say μ ($\mu > 1$), random elementary perturbations, like p_{ij}. Here, the parameter μ is referred to as a reconstruction (mutation) level. For the sake of robustness, we let the parameter μ vary within some interval, say $[\mu_{\min}, \mu_{\max}] \subseteq [2, n]$. The values of $\mu_{\min}, \mu_{\max}$ can be related to the problem size, n, i.e. $\mu_{\min} = \max(2, \lfloor \rho_1 n \rfloor)$ and $\mu_{\max} = \max(2, \lfloor \rho_2 n \rfloor)$, where ρ_1, ρ_2 ($0 < \rho_1 \leq \rho_2 \leq 1$) are user-defined coefficients (reconstruction factors). In our implementation, μ varies in the following way: at the beginning, μ is equal to $\mu_{\min}$;

```
function shift-mutation(π,μ) // shift mutation procedure for the QAP //
  save μ elements of the permutation π starting from the location n–μ+1
  shift n–μ elements to right starting from the location μ
  replace μ elements by the elements previously saved starting from the location 1
  return π
end // shift-mutation //
```

Fig. 5. Template of the shift mutation procedure.

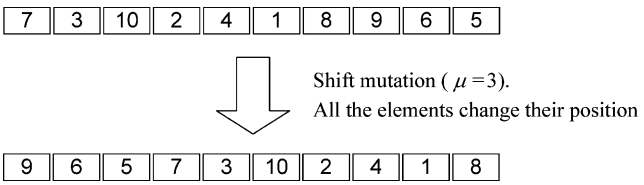


Fig. 6. Example of shift mutation.

once the maximum value $\mu_{\max}$ has been reached (or a better locally optimal solution has been found), the value of μ is immediately dropped to the minimum value $\mu_{\min}$, and so on.

In the limited iterated TS algorithm, we accept the best locally optimal solution as the only candidate for the reconstruction.

The parameter Q_{lits} (the number of iterations) plays the role of the termination criterion for the LITS procedure. Note that the increased value of Q_{lits} is used in two cases: first, when the initial population is being generated; second, when the restart of the algorithm is being performed.

The culling (updating) of the population takes place every time before going to the next generation. After k ($k = Q_{\text{cross}}$) offsprings have been added to the population, it is sorted according to the increasing values of the objective function. Solutions with the greatest objective function value are then removed away from the population to keep the size of the population constant. A special kind of updating is performed when convergence of the algorithm is observed, i.e. further improvements of the individuals are unlikely. As the convergence criterion, a threshold of entropy of the population is used. (For the definition of the entropy, the reader is addressed to Ref. [30].) So, if the entropy, E, is close enough to zero ($E \leq 0.1$), the population is updated in the following way: all the individuals but the best undergo a special mutation—shift mutation (SM); the mutated individuals are then improved by the LITS procedure; after this, IHGA proceeds with the new population in an ordinary way.

Regarding the SM, this mutation was used in Ref. [35], in an efficient TS algorithm. It has been proven to be a highly effective diversification tool. The idea of SM is, nevertheless, very simple: having a permutation just shift all the items to right (left) (in a wrap-around fashion) by a predefined number of positions, μ ($0 < \mu < n$, n is the size of the permutation). SM distinguishes itself for the important property: as long as the population size, PS, is less than the problem size, n, the following equality holds $d(\pi_i, \pi_j) = n, \forall \pi_i, \pi_j \in P$, where P is the current population, π_i, π_j are the mutated permutations, and d is the distance between permutations (see above). The template of the SM procedure is presented in Fig. 5. An example is shown Fig. 6.

The run time of IHGA is controlled by a special parameter—the total number of generations, Q_{gen}, i.e. the algorithm is continued until exactly Q_{gen} generations have been executed.

3.3. Computational results

A large number of computational experiments were carried out in order to test the performance of the proposed algorithm. We used a wide range of the QAP instances from the QAP library QAPLIB [36]. The types of the QAP instances are as follows:

(a) randomly generated 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 instances (instances of this class are real world instances from practical applications, among them: chr25a, els19, esc32a, esc64a, esc128, kra30a, kra30b, ste36a, ste36b, ste36c, tai64c, tai256c (also known as grey_16_16_92); for example, the instances tai64c, tai256 occur in the generation of grey patterns (Section 3.1));
(c) real-life like instances (they are generated in such a way that the entries of the data matrices resemble a distribution from real-life problems; the instances are

Table 1
Comparison of the algorithms on randomly generated instances (type (a))

Instance name	n	BKV	θ_{avg}					t
			RTS	GH	FANT	GAHRR	IHGA	
tai20a	20	703,482	$0.061_{(8)}$	$0.411_{(2)}$	$0.857_{(0)}$	$0.061_{(8)}$	**0**	2.5
tai25a	25	1,167,256	$0.125_{(7)}$	$0.382_{(2)}$	$1.100_{(1)}$	$0.088_{(7)}$	**0**	6.0
tai30a	30	1,818,146	$0.058_{(7)}$	$0.362_{(4)}$	$0.940_{(1)}$	$0.019_{(8)}$	**0**	16
tai35a	35	2,422,002	$0.184_{(4)}$	$0.643_{(0)}$	$1.271_{(0)}$	$0.126_{(6)}$	**0**	36
tai40a	40	3,139,370	$0.436_{(0)}$	$0.618_{(0)}$	$1.394_{(0)}$	$0.338_{(0)}$	$\mathbf{0.209}_{(1)}$	85
tai50a	50	4,941,410	$0.736_{(0)}$	$0.871_{(0)}$	$1.645_{(0)}$	$0.567_{(0)}$	$\mathbf{0.424}_{(1)}$	300
tai60a	60	7,208,572	$0.816_{(0)}$	$1.009_{(0)}$	$1.596_{(0)}$	$0.590_{(0)}$	$\mathbf{0.547}_{(0)}$	720
tai80a	80	13,557,864	$0.611_{(0)}$	$0.593_{(0)}$	$1.249_{(0)}$	$\mathbf{0.271}_{(0)}$	$0.320_{(0)}$	3200
tai100a	100	21,125,314	$0.582_{(0)}$	$0.493_{(0)}$	$1.160_{(0)}$	$0.296_{(0)}$	$\mathbf{0.259}_{(0)}$	12,000

Note: IHGA was run in two variants, first using $(\rho_1, \rho_2) = (0.3, 0.4)$, and second using $(\rho_1, \rho_2) = (0.4, 0.5)$. The results for the best of two variants are displayed in the column IHGA.

Table 2
Comparison of the algorithms on real-life instances (type (b))

Instance name	n	BKV	θ_{avg}					t
			RTS	GH	FANT	GAHRR	IHGA	
chr25a	25	3796	$5.148_{(1)}$	$3.825_{(1)}$	$6.033_{(3)}$	$0.232_{(9)}$	**0**	1.9
els19	19	17,212,548	$3.076_{(2)}$	$0.421_{(9)}$	$0.421_{(9)}$	**0**	**0**	0.1
esc32a	32	130	$1.231_{(3)}$	$0.769_{(6)}$	$2.615_{(3)}$	$0.154_{(9)}$	**0**	2.5
esc64a	64	116	$0.862_{(6)}$	**0**	**0**	**0**	**0**	4.0
esc128	128	64	$16.562_{(1)}$	**0**	**0**	**0**	**0**	24
kra30a	30	88,900	$0.403_{(7)}$	**0**	$1.380_{(2)}$	$0.224_{(8)}$	**0**	1.8
kra30b	30	91,420	$0.023_{(7)}$	$0.031_{(6)}$	$0.071_{(7)}$	$0.028_{(7)}$	**0**	2.3
ste36a	36	9526	$0.063_{(6)}$	$0.086_{(6)}$	$0.355_{(2)}$	$0.061_{(7)}$	**0**	9.8
ste36b	36	15,852	$0.106_{(8)}$	$0.025_{(9)}$	$0.426_{(5)}$	**0**	**0**	2.4
ste36c	36	8239.11	$0.022_{(7)}$	$0.051_{(5)}$	$0.139_{(4)}$	$0.001_{(9)}$	**0**	3.8
tai64c	64	1,855,928	$0.015_{(4)}$	**0**	**0**	**0**	**0**	0.3
tai256c	256	44,759,294[a]	$0.086_{(0)}$	$0.078_{(0)}$	$0.121_{(0)}$	$0.053_{(1)}$	$\mathbf{0.021_{(5)}}$	1280

Note: IHGA was run in three variants, first using $Q_{lits} = 5$, second using $Q_{lits} = 7$, and third with $Q_{lits} = 10$. The results for the best of three variants are displayed in the column IHGA.
[a] Comes from Ref. [37].

Table 3
Comparison of the algorithms on real-life like instances (type (c))

Instance name	n	BKV	θ_{avg}					t
			RTS	GH	FANT	GAHRR	IHGA	
tai20b	20	122,455,319	**0**	$0.045_{(9)}$	$0.091_{(8)}$	**0**	**0**	0.4
tai25b	25	344,355,646	$0.044_{(8)}$	**0**	$0.007_{(9)}$	$0.007_{(9)}$	**0**	0.9
tai30b	30	637,117,113	$0.408_{(3)}$	$0.000_{(9)}$	$0.029_{(7)}$	**0**	**0**	1.8
tai35b	35	283,315,445	$0.233_{(5)}$	$0.132_{(4)}$	$0.198_{(1)}$	$0.059_{(7)}$	**0**	3.0
tai40b	40	637,250,948	$0.204_{(6)}$	**0**	**0**	**0**	**0**	7.0
tai50b	50	458,821,517	$0.239_{(0)}$	$0.035_{(7)}$	$0.222_{(0)}$	$0.002_{(8)}$	**0**	20
tai60b	60	608,215,054	$0.291_{(0)}$	$0.018_{(6)}$	$0.179_{(3)}$	$0.000_{(9)}$	**0**	40
tai80b	80	818,415,043	$0.270_{(0)}$	$0.354_{(2)}$	$0.312_{(0)}$	$0.003_{(2)}$	**0**	150
tai100b	100	1,185,996,137	$0.186_{(0)}$	$0.045_{(3)}$	$0.096_{(0)}$	$0.014_{(3)}$	**0**	440
tai150b	150	498,896,643[a]	$0.392_{(0)}$	$0.394_{(0)}$	$0.514_{(0)}$	$0.200_{(0)}$	$\mathbf{0.111_{(2)}}$	2300

Note: IHGA was run in two variants, first using $Q_{lits} = 5$, and second using $Q_{lits} = 7$. The results for the best of two variants are displayed in the column IHGA.
[a] Comes from Ref. [38].

denoted by tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, tai60b, tai80b, tai100b, tai150b).

For the comparison, we used robust tabu search (RTS) algorithm [27], genetic hybrid (GH) algorithm [30], fast ant system (FANT) [25], and genetic algorithm hybridized with R&R procedure (GAHRR) [33]. These algorithms belong to the most powerful algorithms for the QAP. As a main performance measure, the average deviation from the best-known solution is chosen. The average deviation, θ_{avg}, is defined according to the formula $\theta_{avg} = 100(z_{avg} - z_b)/z_b$ (%), where z_{avg} is the average objective function value over

Table 4
Values of the parameters for the algorithm IHGA

Problem type	PS	Q_{gen}	Q_{cross}	Q_{lits}	τ	SF	(ρ_1, ρ_2)
(a)	$3\sqrt{n}$	$\frac{1}{3}n$	$\frac{2}{3}PS$	3	$\frac{1}{2}n^2$	1.3	Depends on problem
(b)	$3\sqrt{n}$	Depends on problem	$\frac{1}{3}PS$	Depends on problem	n	1.7	(0.35,0.45)
(c)	$2\sqrt{n}$	Depends on problem	$\frac{1}{2}PS$	Depends on problem	n	1.7	(0.35,0.45)

The meaning of the notations used in the table is as follows: PS, the population size; Q_{gen}, # of generations; Q_{cross}, # of crossovers per generation, Q_{lits}, # of LITS iterations; τ, # of TS steps; SF, the selection factor; (ρ_1, ρ_2), the reconstruction factors; n, the size of the instance. Regarding Q_{gen} for the instance types (b) and (c), it varies from $0.1n$ to n.

Table 5
New best-known solutions for grey patterns problems

Instance name	n	Previous best known value	New best known value
grey_16_16_71	256	24,742,496[a]	24,693,608
grey_16_16_72	256	25,570,996[a]	25,529,984
grey_16_16_73	256	26,423,856[a]	26,382,310
grey_16_16_74	256	27,276,468[a]	27,235,240
grey_16_16_75	256	28,154,258[a]	28,114,952
grey_16_16_76	256	29,001,308[a]	29,000,908
grey_16_16_77	256	29,895,152[b]	29,894,452
grey_16_16_78	256	30,820,660[a]	30,797,954
grey_16_16_79	256	31,736,576[a]	31,702,182
grey_16_16_80	256	32,658,876[a]	32,593,088

Problems of this type are described in Refs. [20,38] under the name grey_n_1_n_2_m, where m is the density of the grey ($0 \leq m \leq n = n_1 \times n_2$), and $n_1 \times n_2$ is the size of a frame.

[a] Comes from Ref. [38].

[b] Comes from Ref. [33].

W ($W = 1, 2, \ldots$) restarts (i.e. single applications of the algorithm to a problem instance), and z_b is the best known value (BKV) of the objective function. (BKVs are from Ref. [36].)

The results of the comparison, i.e. the average deviations from BKV, as well as the approximated CPU times per restart (in seconds) are presented in Tables 1–3. The deviations are averaged over 10 restarts. CPU times are given for the x86 Family 6 processor. The best values are printed in bold face (in addition, in parenthesis, we give the number of times that BKV is found (if $\theta_{avg} > 0$)). The values of the parameters of the algorithm IHGA for the different problem types are presented in Table 4.

The values of the control parameters of the remaining algorithms were chosen in such a way that all the algorithms use approximately the same CPU time. Some differences in run time are, in most cases, negligible and cannot influence the results significantly.

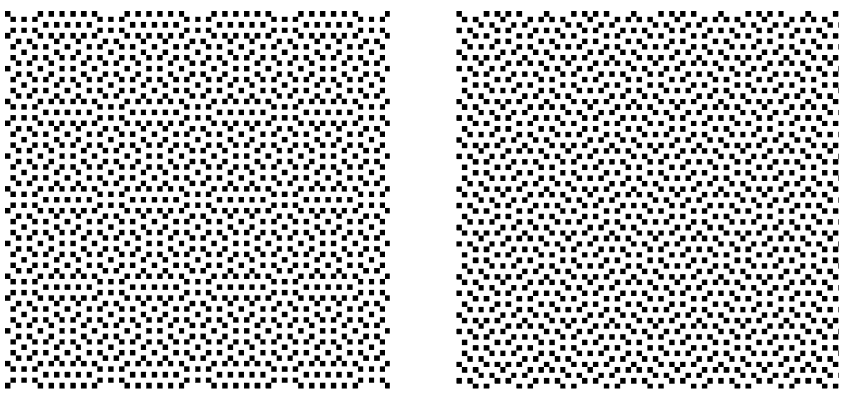
It can be seen from Tables 1–3 that the algorithm IHGA appears to be superior to other efficient algorithms. Our algorithm obtained the best-known solutions for all the instances tested, excepting the large difficult random instances tai60a, tai80a and tai100a. Nevertheless, the results obtained for these instances are considerably better than those of FANT and RTS due to Taillard [25,27] or GH due to Fleurent and Ferland [30]. Moreover, for almost all (excepting the largest available) real-life and real-life like instances, the average deviation of the objective function is equal to zero—this indicates that these instances are most probably solved to pseudo-optimality. It is important that IHGA finds the pseudo-optimal solutions surprisingly quickly; for example, for the instance tai100b, approximately 400 s are enough (see Table 3). It should also be stressed that IHGA discovered new best-known solutions for several large grey patterns problems. New solutions are presented in Table 5. Some visual examples are shown in Fig. 7.

4. Concluding remarks

The method that has been widely applied to many fields of computer science, among them optimization problems, is the genetic approach. In this paper, we introduce an IHGA for the difficult combinatorial optimization problem, the QAP.

In contrast to standard GAs that rely on the concept of biological evolution, the hybrid GAs are based rather on cultural evolution. In hybrid GAs, the individuals (solutions) are improved by using efficient local optimization techniques. In some sense, this is similar to ideas' lifetime—human ideas usually undergo many transformations before passing to the future generations.

The success of a hybrid GA depends, in a high degree, on the efficiency of a local optimization algorithm. In this paper, we propose a very promising procedure based upon so-called LITS approach that, in turn, rely on the TS and the ILS principle. The robustness of LITS is achieved by applying a proper diversification mechanism, i.e. one tries to obtain better optimization results by reconstruction of



(a) (b)

Fig. 7. Examples of grey frames of density (a) 71/256 and (b) 72/256 obtained by solving QAP by IHGA.

a locally optimal solution and a following TS-relying improvement procedure.

It is important that the LITS procedure includes the reconstructions of solutions: these reconstructions can be viewed as mutations, so this helps greatly in the case of incorporating LITS into the hybrid GA. In our improved HGA, the populations are typically very compact—this is due to the high performance of the LITS procedure. Therefore, a lot of computations can be saved when comparing to other GAs. Another favourable feature of our GA is related to a restart mechanism. This mechanism is distinguished for a special sort of mutations, so-called SMs, which have been proven to be extremely robust and effective.

All these features coupled with additional refinements furthered the development of a powerful hybrid GA for the QAP. The results from the experiments show that this algorithm appears to be superior to other intelligent optimization techniques, as well as the GA that was presented by the author at ES2002 (Cambridge, UK, 2002). For many QAP instances from QAPLIB (especially, for the real life and real-life like instances), a few seconds are enough for our algorithm to find the best known (pseudo-optimal) solutions. Moreover, for several grey patterns problems (that are special QAP cases), record-breaking solutions were obtained.

Acknowledgements

Author is grateful to Prof. E. Taillard for providing the codes of the robust tabu search and fast ant algorithms. He also would like to thank Prof. C. Fleurent for the paper that helped coding the hybrid genetic algorithm.

References

[1] R. Chelouah, P. Siarry, A continuous genetic algorithm designed for the global optimization of multimodal functions, J. Heurist. 6 (2000) 191–213.
[2] T.N. Bui, B.R. Moon, Genetic algorithm and graph partitioning, IEEE Trans. Comput. 45 (1996) 841–855.
[3] Z. Drezner, S. Salhi, Using metaheuristics for the one-way and two-way network design problem, Nav. Res. Logist. 49 (2002) 449–463.
[4] D.M. Miller, H.C. Chen, J. Matson, Q. Liu, A hybrid genetic algorithm for the single machine scheduling problem, J. Heurist. 5 (1999) 437–454.
[5] M. Yagiura, T. Ibaraki, Genetic and local search algorithms as robust and simple optimization tools, in: I.H. Osman (Ed.), Meta-heuristics: Theory and Applications, Kluwer, Dordrecht, 1996, pp. 63–82.
[6] J.E. Beasley, P.C. Chu, A genetic algorithm for the set covering problem, Eur. J. Oper. Res. 94 (1996) 392–404.
[7] P. Merz, B. Freisleben, Genetic local search for the TSP: new results, Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, Indianapolis, USA, 1997.
[8] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[9] L. Davis, Handbook of Genetic Algorithms, Van Nostrand, New York, 1991.
[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[11] H. Mühlenbein, Genetic algorithms, in: E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 137–171.
[12] P. Moscato, Memetic algorithms: a short introduction, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, 1999, pp. 219–234.
[13] H.R. Lourenco, O. Martin, T. Stützle, Iterated local search, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, Norwell, 2002, pp. 321–353.
[14] N. Mladenović, P. Hansen, Variable neighbourhood search, Comput. Oper. Res. 24 (1997) 1097–1100.
[15] G. Schrimpf, K. Schneider, H. Stamm-Wilbrandt, V. Dueck, Record breaking optimization results using the ruin and recreate principle, J. Comput. Phys. 159 (2000) 139–171.
[16] L. Eshelman, The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination, in: G.J.E. Rowlings (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991, pp. 265–283.
[17] F. Glover, M. Laguna, Tabu Search, Kluwer, Dordrecht, 1997.
[18] J.W. Dickey, J.W. Hopkins, Campus building arrangement using TOPAZ, Transp. Res. 6 (1972) 59–68.
[19] T.C. Hu, E.S. Kuh (Eds.), VLSI Circuit Layout: Theory and Design, IEEE Press, New York, 1985.
[20] E. Taillard, Comparison of iterative searches for the quadratic assignment problem, Location Sci. 3 (1995) 87–105.
[21] J. Krarup, P.M. Pruzan, Computer-aided layout design, Math. Program. Study 9 (1978) 75–94.
[22] R.E. Burkard, J. Offermann, Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme, Z. Oper. Res. 21 (1977) 121–132.
[23] R.E. Burkard, E. Çela, P.M. Pardalos, L. Pitsoulis, The quadratic assignment problem, in: D.Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, vol. 3, Kluwer, Dordrecht, 1998, pp. 241–337.
[24] S. Sahni, T. Gonzalez, P-complete approximation problems, J. ACM 23 (1976) 555–565.
[25] E. Taillard, FANT: fast ant system, Tech. Report IDSIA-46-98, Lugano, Switzerland, 1998.
[26] A. Bölte, U.W. Thonemann, Optimizing simulated annealing schedules with genetic programming, Eur. J. Oper. Res. 92 (1996) 402–416.
[27] E. Taillard, Robust taboo search for the QAP, Parallel Comput. 17 (1991) 443–455.
[28] R.K. Ahuja, J.B. Orlin, A. Tiwari, A greedy genetic algorithm for the quadratic assignment problem, Comput. Oper. Res. 27 (2000) 917–934.
[29] Z. Drezner, A new genetic algorithm for the quadratic assignment problem, INFORMS J. Comput. 15 (2003) 320–330.
[30] C. Fleurent, J.A. Ferland, Genetic hybrids for the quadratic assignment problem, in: 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, pp. 173–188.
[31] M.H. Lim, Y. Yuan, S. Omatu, Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem, Comput. Optim. Appl. 15 (2000) 249–268.
[32] P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, IEEE Trans. Evol. Comput. 4 (2000) 337–352.
[33] A. Misevicius, Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem, in: M. Bramer, A. Preece, F. Coenen (Eds.), Research and Development in Intelligent Systems XIX, Proceedings of 22nd SGAI International

Conference on Knowledge Based Systems and Applied Artificial Intelligence, Cambridge, UK (2002), Springer, London, 2002, pp. 163–176.
[34] D.M. Tate, A.E. Smith, A genetic approach to the quadratic assignment problem, Comput. Oper. Res. 1 (1995) 73–83.
[35] A. Misevicius, A tabu search algorithm for the quadratic assignment problem, Working Paper, Kaunas University of Technology, Lithuania, 2002, (under review).
[36] R.E. Burkard, S. Karisch, F. Rendl, QAPLIB-a quadratic assignment problem library, J. Glob. Optim. 10 (1997) 391–403.
[37] T. Stützle, MAX-MIN ant system for quadratic assignment problems, Res. Report AIDA-97-04, Darmstadt University of Technology, Germany, 1997.
[38] E. Taillard, L.M. Gambardella, Adaptive memories for the quadratic assignment problem, Tech. Report IDSIA-87–97, Lugano, Switzerland, 1997.