Ruin and Recreate Principle Based Approach for the Quadratic Assignment Problem

Alfonsas Misevicius

Kaunas University of Technology, Department of Practical Informatics,
Studentu St. 50–400a, LT–3031 Kaunas, Lithuania
`misevi@soften.ktu.lt`

Abstract. In this paper, we propose an algorithm based on so-called ruin and recreate (R&R) principle. The R&R approach is conceptual simple but at the same time powerful meta-heuristic for combinatorial optimization problems. The main components of this method are a ruin (mutation) procedure and a recreate (improvement) procedure. We have applied the R&R principle based algorithm for a well-known combinatorial optimization problem, the quadratic assignment problem (QAP). We tested this algorithm on a number of instances from the library of the QAP instances – QAPLIB. The results obtained from the experiments show that the proposed approach appears to be significantly superior to a "pure" tabu search on real-life and real-life like QAP instances.

1 Introduction

The quadratic assignment problem (QAP) is 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 permutations of the integers from 1 to n be given. Find a permutation $\pi = (\pi(1), \pi(2), ..., \pi(n)) \in \Pi$ that minimizes

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

The context in which Koopmans and Beckmann [1] first formulated this problem was the facility location problem. In this problem, one is concerned with locating n units on n sites with some physical products flowing between the facilities, and with distances between the sites. The element a_{ij} is the "flow" from unit i to unit j, and the element b_{kl} represents the distance between the sites k and l. The permutation $\pi = (\pi(1), \pi(2), ..., \pi(n))$ can be interpreted as an assignment of units to sites ($\pi(i)$ denotes the site what unit i is assigned to). Solving the QAP means searching for an assignment π that minimizes the "transportation cost" z.

An important area of the application of the QAP is computer-aided design (CAD), more precisely, the placement of electronic components into locations (positions) on a board (chip) [2,3,4]. Other applications of the QAP are: campus planning [5], hospital layout [6], image processing (design of grey patterns) [7], testing of electronic devices [8], typewriter keyboard design [9], etc (see [10,11] for a more detailed description of the QAP applications).

The quadratic assignment problem is one of the most complex combinatorial optimization problems. It has been proved that the QAP is NP-hard [12]. To this date problems of size n, where n>30, are not practically solvable in terms of obtaining exact solutions. Therefore, heuristic approaches have to be used for solving medium- and large-scale QAPs: ant algorithms [13,14], genetic algorithms [15,16,17], greedy randomized search (GRASP) [18], iterated local search [19], simulated annealing [20,21], tabu search [22,23,24]. An extended survey of heuristics for the QAP one can find in [11,25].

This paper is organized as follows. In Section 2, basic definitions are given. Section 3 outlines the ruin and recreate approach for combinatorial optimization problems. Section 4 describes the R&R principle based algorithm for the quadratic assignment problem. The results of the computational experiments on the QAP instances are presented in Section 5. Section 6 completes the paper with conclusions.

2 Preliminaries

Let S be a set of solutions (solution space) of a combinatorial optimization problem with objective function f: $S \to R$. Furthermore, let N: $S \to 2^S$ be a neighbourhood function which defines for each $s \in S$ a set $N(s) \subseteq S$ – a set of neighbouring solutions of s. Each solution $s' \in N(s)$ can be reached from s by an operation called a move.

Suppose, $S = \{s \mid s = (s(1), s(2), ..., s(n))\}$, where n is the cardinality of the set, i.e. the problem size. Given a solution s from S, a λ-exchange neighbourhood function $N_\lambda(s)$ is defined as follows:

$$N_\lambda(s) = \{s' \mid s' \in S, d(s,s') \leq \lambda\} \quad (2 \leq \lambda \leq n), \tag{2}$$

where $d(s,s')$ is the "distance" between solutions s and s':

$$d(s,s') = \sum_{i=1}^{n} \operatorname{sgn} \mid s(i) - s'(i) \mid .$$

For the QAP, the commonly used case is $\lambda = 2$, i.e. the 2-exchange function N_2. In this case, a transformation from the current permutation (solution) π to the neighbouring permutation π' can formally be defined by using a special move (operator) – 2-way perturbation p_{ij}: $\Pi \to \Pi$ (i,j = 1, 2, ..., n), which exchanges ith and jth elements in the current permutation. The notation $\pi' = \pi \oplus p_{ij}$ means that π' is obtained from π by applying p_{ij}. The difference in the objective function values (when moving from the current permutation to the neighbouring one) can be calculated in $O(n)$ operations:

$$\begin{aligned}\Delta_z(\pi,i,j) = {} & (a_{ij} - a_{ji})(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)}) + \\ & \sum_{k=1,k\neq i,j}^{n} \left[(a_{ik} - a_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) + (a_{ki} - a_{kj})(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)})\right],\end{aligned} \tag{3}$$

where a_{ii}(or b_{ii}) = const, $\forall i \in \{1, 2, ..., n\}$.

If the matrix $\boldsymbol{A}$ and/or matrix $\boldsymbol{B}$ are symmetric formula (3) becomes much more simpler. For example, if the matrix $\boldsymbol{B}$ is symmetric, one can transform the matrix $\boldsymbol{A}$ to the symmetric one $\boldsymbol{A}'$ by adding up corresponding entries of $\boldsymbol{A}$.

Formula (3) then reduces to the following formula

$$\Delta_z(\pi,i,j) = \sum_{k=1,k\neq i,j}^{n} (a'_{ik} - a'_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}), \tag{4}$$

where $a'_{ik} = a_{ik} + a_{ki}$, $\forall i,k \in \{1, 2, ..., n\}$, $i \neq k$. Moreover, for two consecutive solutions π and $\pi' = \pi \oplus p_{uv}$, if all values $\Delta_z(\pi,i,j)$ have been stored, the values $\Delta_z(\pi',i,j)$ ($i \neq u,v$ and $j \neq u,v$) can be computed in time $O(1)$ [24]:

$$\begin{aligned}\Delta_z(\pi',i,j) = \Delta z(\pi,i,j) + (a_{iu} - a_{iv} + a_{jv} - a_{ju})(b_{\pi(i)\pi(u)} - b_{\pi(i)\pi(v)} + b_{\pi(j)\pi(v)} - b_{\pi(j)\pi(u)}) +\\ (a_{ui} - a_{vi} + a_{vj} - a_{uj})(b_{\pi(u)\pi(i)} - b_{\pi(v)\pi(i)} + b_{\pi(v)\pi(j)} - b_{\pi(u)\pi(j)}).\end{aligned} \tag{5}$$

(If $i = u$ or $i = v$ or $j = u$ or $j = v$, then formula (3) is applied.)

3 Ruin and Recreate Principle

The ruin and recreate (R&R) principle was formulated by Schrimpf et al. [26]. Note also that this principle has some similarities with other approaches, among them: combined local search heuristic (chained local optimization) [27], iterated local search [19], large step Markov chains [28], variable neighbourhood search [29]. As mentioned in [26], the basic element of the R&R principle is to obtain better optimization results by a reconstruction (destruction) of an existing solution and a following improvement (rebuilding) procedure. By applying this type of process frequently, i.e. in an iterative way, one seeks for high quality solutions. In the first phase of the process, one reconstructs a significant part of the existing solution, roughly speaking, one "ruins" the current solution. (That is a relatively easy part of the method.) In the second phase, one tries to rebuild the solution just "ruined" as best as one can. Hopefully, the new solution is better than the solution(s) obtained in the previous phase(s) of the improvement. (Naturally, the improvement is the harder part of the method.) There are a lot of different ways to reconstruct (ruin) and, especially, to improve the solutions. So, we think of the ruin and recreate approach as a meta-heuristic – not a pure heuristic.

It should be noted that the approach we have outlined differs a little from the original R&R version. The main distinguishing feature is that, in our version of R&R, there is considered rather an iterative process of some kind of mutations (as reconstructions) and local searches (as improvements) applied to solutions. In the original R&R, in contrast, one speaks rather of a sequence of disintegrations and constructions of solutions (see Schrimpf et al. [26]).

The advantage of the R&R approach – which is very similar to the iterated local search [19] – over the well-known random multistart method (that is based on multiple starts of local improvements applied to randomly generated solutions) is that, instead of generating (constructing) new solutions from scratch, a better idea is to reconstruct (a part of) the current solution: continuing search from this reconstructed solution may allow to escape from a local optimum and to find better solutions. (So, improvements of the reconstructions of local optima are considered rather than improvements of the purely random solutions.) For the same computation time, many more reconstruction improvements can be done than when starting from randomly generated solutions, because the reconstruction improvement procedure requires only a few steps to reach the next local optimum.

The R&R principle can be thought of as a method with so-called "large" moves (moves consisting of a large number of perturbations) [26], instead of "small" moves (e.g. pair-wise interchanges) that typically take place in "classical" algorithms. For simple problems, there is no need to use "large" moves – "small" moves are enough. The "large" moves – that are also referred to as "kick" moves [27] – are very important when dealing with complex problem (QAP among them). These problems can often be seen as "discontinuous": if one walks in the solution space, the qualities of the solutions can be significantly different, i.e. the "landscapes" of these problems can be very rugged.

Two aspects regarding to the "kick" moves are of high importance (see also [19]). Firstly, the "kick" move should be large (strong) enough to allow to leave the current locally optimal solution and to enable the improvement procedure to find, probably, better solution. However, if the "kick" move is too large, the resulting algorithm might be quite similar to random multistart, which is known to be not a very efficient algorithm. Secondly, the "kick" move should be small (weak) enough to keep features (characteristics) of the current local minimum since parts of the solution may be close to the ones of the globally optimal solution. However, if the "kick" move is too small, the improvement procedure may return to the solution to which the "kick" move has been applied, i.e. a cycling may occur.

In the simplest case, it is sufficient to use for the "kick" move a certain number of random perturbations (further, we shall refer those perturbations to as mutation). Doing so can be interpreted as a random search of higher order neighbourhoods N_λ (λ>2), i.e. random variable neighbourhood search [29].

Additional component of the R&R method is an acceptance criterion that is used to decide which solution is to be chosen for the "ruin". Here, two main alternatives are so-called intensification (exploitation) and diversification (exploration). Intensification is achieved by choosing only the best local optimum as a candidate for the "ruin". On the other hand, diversification takes place if every new local optimum is accepted for the reconstruction.

The paradigm of the ruin and recreate approach, which is conceptually surprisingly simple, is presented in Fig. 1.

4 Ruin and Recreate Principle Based Algorithm for the QAP

All we need by creating the ruin and recreate principle based algorithm for a specific problem is to design four components: 1) an initial solution generation (construction) procedure, 2) a solution improvement procedure (we shall also refer this procedure to as local search procedure), 3) a solution reconstruction (mutation) procedure, and 4) a candidate acceptance (choosing) rule.

Now we present details of the ruin and recreate principle based algorithm for the QAP, which is entitled R&R-QAP.

```
procedure R&R { ruin and recreate procedure }
   generate (or construct) initial solution $s^\circ$
   $s^\bullet$ := recreate($s^\circ$) { recreate (improve) the initial solution }
   $s^* := s^\bullet$, $s := s^\bullet$
   repeat { main loop }
      $s$ := choose_candidate_for_ruin($s$,$s^\bullet$)
      $s^{\sim}$ := ruin($s$)        { ruin (reconstruct) the current solution }
      $s^\bullet$ := recreate($^{\sim}$)   { recreate (improve) the ruined solution }
      if $s^\bullet$ is better than $s^*$ then $s^* := s^\bullet$
   until termination criterion is satisfied
   return $s^*$
end { R&R }
```

Fig. 1. The paradigm of the ruin and recreate (R&R) principle based procedure

4.1 Initial Solution Generation

We use randomly generated permutations as initial permutations for the algorithm R&R-QAP. These permutations can be generated by a very simple procedure.

4.2 Local Search

In principle, any local search concept based algorithm can be applied at the improvement phase of R&R method. In the simplest case, a greedy descent ("first improvement") algorithm or steepest descent ("best improvement") algorithm (also known as a "hill climbing") can be used. Yet, it is possible to apply more sophisticated algorithms, like limited simulated annealing or tabu search. In our algorithm, we use the tabu search algorithm, more precisely, a modified version of the robust tabu search algorithm due to Taillard [24]. The framework of the algorithm can briefly be described as follows. Initialize tabu list, $\boldsymbol{T} = (t_{ij})_{n\times n}$, and start from an initial solution π. Continue the following process until a termination criterion is satisfied (a predetermined number of trials is executed): 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

$N_2'(\pi) = \{\pi' \mid \pi' \in N_2(\pi), (\pi' = \pi \oplus p_{ij}$ and p_{ij} is not tabu) or $z(\pi') < z(\pi''')\}$ (π''' is the best so far solution); b) replace the current solution π by the neighbour π'' (even if $z(\pi'') - z(\pi) > 0$), and use as a starting point for the next trials; c) update the tabu list $\boldsymbol{T}$. The last found solution, which is locally optimal, is declared as a solution of the tabu search algorithm. The detailed template of the tabu search based local search algorithm for the QAP is presented in Fig. 2.

```
function local_search($\pi$,$n$,$\tau$) { local search (based on tabu search) for the QAP }
  { $\pi$ – the current permutation, $n$ – problem size, $\tau$ – number of iterations }
  set lower and higher tabu sizes $h_{min}$ and $h_{max}$
  $\pi^{\bullet} := \pi$ { $\pi^{\bullet}$ denotes the best permutation found }
  calculate the objective function differences $\delta_{ij} = \Delta_z(\pi,i,j)$: $i = \overline{1,n-1}$; $j = \overline{i+1,n}$
  $T := 0$
  choose $h$ randomly between $h_{min}$ and $h_{max}$
  $i := 1, j := 1, k := 1$
  improved := FALSE
  while ($k \le \tau$) or improved do begin { main loop }
    $\delta_{min} := \infty$ { $\delta_{min}$ – minimum difference in the objective function values }
    for $l := 1$ to $|N_2|$ do begin
      $i := \mathrm{if}(j < n, i, \mathrm{if}(i < n-1, i+1, 1))$, $j := \mathrm{if}(j < n, j+1, i+1)$
      tabu := if($t_{ij} \ge k$,TRUE,FALSE), aspired := if($z(\pi) + \delta_{ij} < z(\pi^{\bullet})$,TRUE,FALSE)
      if (($\delta_{ij} < \delta_{min}$) and not tabu) or aspired then begin
        $\delta_{min} := \delta_{ij}$, $u := i$, $v := j$
      end { if }
    end { for $l$ }
    if $\delta_{min} < \infty$ then begin
      $\pi := \pi \oplus p_{uv}$ { replace the current permutation by the new one }
      if $z(\pi) < z(\pi^{\bullet})$ then $\pi^{\bullet} := \pi$
      improved := if($\delta_{uv} < 0$,TRUE,FALSE)
      update the differences $\delta_{lm}$: $l = \overline{1,n-1}$; $m = \overline{l+1,n}$
      $t_{uv} := k + h$
      if $k$ mod $2h_{max} = 0$ then choose new $h$ randomly between $h_{min}$ and $h_{max}$
    end { if }
  end { while }
  return $\pi = \pi^{\bullet}$
end { local_search }
```

Fig. 2. Template of the tabu search based local search algorithm for the QAP.
Notes. 1. The objective function difference δ_{ij} is calculated according to (3) (or 4) formula.
2. The difference δ_{lm} is updated according to (5) formula.
3. The function if(x,y_1,y_2) returns y_1 if x=TRUE, otherwise it returns y_2

4.3 Mutation

As mentioned in Section 3, a mutation of the current solution is achieved by performing a complex move, i.e. a move in the neighbourhood N_λ, where $\lambda>2$. This can be modelled by generating $\mu=\lfloor(\lambda+1)/2\rfloor$ sequential elementary perturbations, like p_{ij}; here, the parameter μ ($\mu\ge2$) is referred to as a mutation level.

```
function $mutation(\pi,n,\mu)$ { mutation procedure for the QAP }
  { $\pi$ – the current permutation, $n$ – problem size, $\mu$ – mutation level ($2\le\mu\le n$) }
  $\pi^{\bullet} := \pi$
  for $k := 1$ to $\mu$ do begin { main loop }
    $i := randint(1,n)$, $j := randint(1,n-1)$
    if $i\le j$ then $j := j + 1$
    $\pi := \pi \oplus p_{ij}$ { replace the current permutation by the new one }
    if $z(\pi)<z(\pi^{\bullet})$ then break
  end { for $k$ }
  return $\pi$
end { $mutation$ }
```

Fig. 3. Template of the mutation procedure for the QAP.
Note. The function randint(x,y) returns integer random, uniformly distributed number between x and y

```
procedure $R\&R\text{-}QAP$ { ruin and recreate principle based algorithm for the QAP }
  { input: $A,B$ – flow and distance matrices, $n$ – problem size }
  { $Q$ – the total number of iterations }
  { $\alpha$, $\beta_{min}$, $\beta_{max}$ – the control parameters ($\alpha>0$, $0<\beta_{min}\le\beta_{max}\le1$) }
  { output: $\pi^*$ – the best permutation found }
  $\tau := \max(1,\lfloor\alpha n\rfloor)$, $\mu_{min} := \max(2,\lfloor\beta_{min}n\rfloor)$, $\mu_{max} := \max(2,\lfloor\beta_{max}n\rfloor)$
  generate random initial permutation $\pi^{\circ}$
  $\pi^{\bullet} := local\_search(\pi^{\circ},n,\tau)$
  $\pi^* := \pi^{\bullet}$, $\pi := \pi^{\bullet}$
  $\mu := \mu_{min} - 1$ { $\mu$ is the current value of the mutation level }
  for $q := 1$ to $Q$ do begin { main loop of $R\&R\text{-}QAP$ }
    $\pi := \text{if}(z(\pi^{\bullet}) < z(\pi), \pi^{\bullet}, \pi)$ { $\pi$ is the candidate for the mutation }
    $\mu := \text{if}(\mu < \mu_{max}, \mu + 1, \mu_{min})$
    $\tilde{\pi} := mutation(\pi,n,\mu)$          { mutate (reconstruct) the permutation $\pi$ }
    $\pi^{\bullet} := local\_search(\tilde{\pi},n,\tau)$     { try to improve the mutated permutation }
    if $z(\pi^{\bullet})<z(\pi^*)$ then begin
      $\pi^* := \pi^{\bullet}$, $\mu := \mu_{min} - 1$ { save the best permutation, reset the mutation level }
    end { if }
  end { for $q$ }
  return $\pi^*$
end { $R\&R\text{-}QAP$ }
```

Fig. 4. Template of the ruin and recreate (R&R) principle based algorithm for the QAP

We can add more robustness to the R&R method if we let vary the parameter μ in some interval, say $[\mu_{\min},\mu_{\max}] \subseteq [2,n]$ (n is the problem size), during the execution of the algorithm. Two ways are possible when varying μ: random and deterministic. In the first case, "a coin is tossed"; in the second one, μ is varied sequentially within the interval $[\mu_{\min},\mu_{\max}]$ starting from $\mu_{\min}$. Once maximum value $\mu_{\max}$ has been reached (or a better locally optimum solution has been found), the value of μ is dropped to the minimum value $\mu_{\min}$, and so on. The last case is used in our implementation. The detailed template of the mutation procedure is presented in Fig. 3.

4.4 Candidate Acceptance

In our algorithm, we accept only the best locally optimal solution as a candidate for the mutation. The candidate solution at qth iteration is defined according to the following formula $\pi^{(q)} = \text{if}\left(z(\pi^{\bullet}) < z(\pi^{(q-1)}),\pi^{\bullet},\pi^{(q-1)}\right)$, where $\pi^{(q)}$ is the candidate solution at the current iteration, $\pi^{(q-1)}$ is the solution before the last local search execution (the best so far solution), and $\pi^{\bullet}$ is the solution after the current local search execution.

The template of the resulting R&R approach based algorithm is presented in Fig. 4. Note that, for convenience, the parameters α>0, $\beta_{\min}$, $\beta_{\max} \in (0,1]$ are used instead of τ, $\mu_{\min}$, $\mu_{\max}$, respectively.

5 Computational Experiments and Results

We have carried out a number of computational experiments in order to test the performance of the algorithm R&R-QAP. The following types of the QAP instances taken from the quadratic assignment problem library QAPLIB [30] were used:

a) real-life instances (instances of this class are real world instances from practical applications of the QAP, among them: chr25a, els19, esc32a, esc64a, kra30a, ste36a, tai64c (also known as grey_8_8_13 [7]));

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

For the comparison, we used the tabu search algorithm due to Taillard [24] entitled as the robust tabu search algorithm – RTS-QAP, because it is among the best algorithms for the quadratic assignment problem. As a performance measure, the average deviation from the best known solution is chosen. The average deviation, θ_{avg}, is defined according to the formula $\theta_{\text{avg}} = 100(z_{\text{avg}} - z_b)/z_b\ [\%]$, where z_{avg} is the average objective function value over W restarts of the algorithm, and z_b is the best known value (BKV) of the objective function. (Note: BKVs are from [30].)

All the experiments were carried out on x86 Family 6 processor. The computations were organized in such a way that both algorithms use identical initial assignments, and require the similar CPU times. The execution of the algorithms is controlled by a fixed a priori number of iterations. Some differences of the run time for RTS-QAP and R&R-QAP are due to non-deterministic behaviour of the local search procedure used in R&R-QAP. The results of the comparison, i.e. the average deviations from BKV for both RTS-QAP and R&R-QAP, as well as approximated CPU times per restart (in seconds) are presented in Table 1. The deviations are averaged over 100 restarts. The best values are printed in bold face.

Table 1. Comparison of the algorithms on real-life and real-life like instances (Q =50, α= 0.1)

Instance name	n	BKV	θ_{avg}				t_{avg}
			RTS-QAP	R&R-QAP[a]	R&R-QAP[b]	R&R-QAP[c]	
chr25a	25	3796	21.198	**16.596**	18.891	20.874	*0.05*
els19	19	17212548	22.765	0.035	0.123	**0.018**	*0.03*
esc32a	32	130	9.692	**7.123**	7.862	8.477	*0.09*
esc64a	64	116	0.966	**0.000**	**0.000**	**0.000**	*0.50*
kra30a	30	88900	2.692	2.422	2.272	**2.247**	*0.09*
ste36a	36	9526	3.562	2.561	**2.456**	2.771	*0.16*
tai20b	20	122455319	14.258	0.261	0.263	**0.226**	*0.03*
tai25b	25	344355646	14.251	0.744	0.419	**0.341**	*0.06*
tai30b	30	637117113	13.519	1.112	1.093	**0.949**	*0.10*
tai35b	35	283315445	8.014	0.799	0.715	**0.563**	*0.16*
tai40b	40	637250948	10.031	1.459	1.492	**1.440**	*0.27*
tai50b	50	458821517	6.741	0.655	0.719	**0.631**	*0.50*
tai60b	60	608215054	7.026	1.143	**1.050**	1.080	*0.90*
tai80b	80	818415043	5.681	**1.360**	1.434	1.589	*2.3*
tai100b	100	1185996137	4.696	0.910	0.853	**0.747**	*5.0*
tai64c	64	1855928	0.410	0.002	**0.000**	0.003	*0.55*

[a] $\beta_{min} = 0.35$, $\beta_{max} = 0.45$; [b] $\beta_{min} = 0.40$, $\beta_{max} = 0.50$; [c] $\beta_{min} = 0.45$, $\beta_{max} = 0.55$.

Looking at Table 1, it turns out that the efficiency of the algorithms depends on the type of the instances (problems) being solved. For some real-life instances, R&R-QAP produces only slightly better results than RTS-QAP. For the real-life like problems, the situation is different: for these problems, R&R-QAP is much more better than RTS-QAP. The results for the particular instances differ dramatically (see, for example, the results for the instances tai20b, tai25b, tai35b).

Two directions that may lead to the improvement of the results of R&R-QAP are as follows: the parameter-improvement-based direction, and the algorithm's-idea-improvement-based direction. First of all, the results can be improved by a more accurate tuning the control parameters α, β_{min} and β_{max}. This can be seen from Table 1: by the proper choosing the values β_{min}, β_{max} the average deviation can be lowered noticeably (see, for example, the results for chr25a, els19, tai25b, tai35b). The results can also be improved by increasing the value of the parameter Q, but at the cost of a longer computation time. Some results for the instances tai*b are shown in Table 2.

Table 2. Additional computational results of R&R-QAP (α=0.01, $\beta_{\min}$=0.45, $\beta_{\max}$=0.55). The following notations are used: W – the total number of restarts, $W^{1\%}$ – the number of restarts at which the solution that is within 1% optimality was found, W^{*} – the number of restarts at which best known solution was found, t_{avg} – the average CPU time per restart (in seconds)

Instance name	R&R-QAP$_{(Q=1000)}$			R&R-QAP$_{(Q=10000)}$			R&R-QAP$_{(Q=100000)}$		
	θ_{avg}	$W/W^{1\%}/W^{*}$	t_{avg}	θ_{avg}	$W/W^{1\%}/W^{*}$	t_{avg}	θ_{avg}	$W/W^{1\%}/W^{*}$	t_{avg}
tai20b	**0.000**	30/**30/30**	*0.17*		————			————	
tai25b	**0.000**	30/**30/30**	*0.36*		————			————	
tai30b	**0.000**	30/**30/30**	*0.67*		————			————	
tai35b	0.083	30/**30**/19	*1.1*	0.037	30/**30**/24	*10.7*	**0.000**	30/**30/30**	*107*
tai40b	0.134	30/28/27	*1.7*	**0.000**	30/**30/30**	*17.0*		————	
tai50b	0.224	30/**30**/8	*3.5*	0.169	30/**30**/6	*34.9*	0.027	30/**30**/28	*349*
tai60b	0.587	30/20/5	*6.2*	**0.000**	30/**30/30**	*62.4*		————	
tai80b	0.598	30/25/0	*16.4*	0.432	30/29/5	*164*	0.198	30/**30**/19	*1640*
tai100b	0.195	30/**30**/0	*35.6*	0.121	30/**30**/5	*356*	0.086	30/**30**/10	*3560*

It should be stressed that R&R-QAP was successful in finding a large number of new best solutions for so-called problems of grey densities. Problems of this type are described in [7,31] under the name grey_n_1_n_2_m, where m is the density of the grey ($0 \le m \le n = n_1 \cdot n_2$), and $n_1 \cdot n_2$ is the size of a frame. New solutions for the instance family grey_16_16_* are presented in Table 3.

Table 3. New best known solutions for grey density problems

Instance name	Previous best known value[a]	New best known value	Instance name	Previous best known value[a]	New best known value
grey_16_16_25	2216160	2215714	**grey_16_16_57**	14798834	14793682
grey_16_16_43	7795062	7794422	**grey_16_16_58**	15395948	15363628
grey_16_16_44	8219428	8217264	**grey_16_16_61**	17202312	17194812
grey_16_16_45	8677670	8676802	**grey_16_16_62**	17824828	17822806
grey_16_16_46	9134172	9130426	**grey_16_16_65**	19859448	19848790
grey_16_16_49	10523016	10518838	**grey_16_16_66**	20655396	20648754
grey_16_16_51	11517060	11516840	**grey_16_16_67**	21461130	21439396
grey_16_16_52	12019082	12018388	**grey_16_16_68**	22271676	22234020
grey_16_16_54	13098850	13096646	**grey_16_16_69**	23086332	23049732
grey_16_16_55	13668486	13661614	**grey_16_16_70**	23898390	23852796
grey_16_16_56	14238840	14229492			

[a] comes from [31].

6 Conclusions

The quadratic assignment problem is a very difficult combinatorial optimization problem. In order to obtain satisfactory results in a reasonable time, heuristic algorithms are to be applied. One of them, a ruin and recreate principle based algorithm (R&R-QAP) is described in this paper. The proposed algorithm applies a reconstruction (mutation) to the best solution found so far and subsequently applies an improvement (local search) procedure. This scheme – despite its conceptual simplicity, easy programming and practical realization – allowed to achieve very good results with small amount of the computation time. The results from the comparison with the robust tabu search algorithm ("pure" tabu search algorithm) show that R&R-QAP produces much more better results than this algorithm (one of the most efficient heuristics for the QAP), as far as real-life and real-life like problems are concerned. The power of R&R-QAP is also corroborated by the fact that many new best known solutions were found for the largest available QAP instances, so-called grey density problems.

There are several possible directions to try to improve the performance of the proposed algorithm: a) implementing a faster local search procedure; b) investigating new reconstruction (mutation) operators; c) introducing a memory based mechanism when selecting the solutions (candidates) for the reconstruction; d) incorporating R&R-QAP into hybrid genetic algorithms as an initial population generation and/or local improvement procedure.

References

1. Koopmans, T., Beckmann, M.: Assignment Problems and the Location of Economic Activities. Econometrica 25 (1957) 53–76
2. Hanan, M., Kurtzberg, J.M.: Placement Techniques. In: Breuer, M.A. (ed.): Design Automation of Digital Systems: Theory and Techniques, Vol.1. Prentice-Hall (1972) 213–282
3. Hu, T.C., Kuh, E.S. (ed.): VLSI Circuit Layout: Theory and Design. IEEE Press, New York (1985)
4. Steinberg, L.: The Backboard Wiring Problem: A Placement Algorithm. SIAM Review 3 (1961) 37–50
5. Dickey, J.W., Hopkins, J.W.: Campus Building Arrangement Using TOPAZ. Transportation Research 6 (1972) 59–68
6. Elshafei, A.N.: Hospital Layout as a Quadratic Assignment Problem. Operations Research Quarterly 28 (1977) 167–179
7. Taillard, E.: Comparison of Iterative Searches for the Quadratic Assignment Problem. Location Science 3 (1995) 87–105
8. Eschermann, B., Wunderlich, H.J.: Optimized Synthesis of Self-testable Finite State Machines. 20th International Symposium on Fault-Tolerant Computing (FFTCS 20) (1990)
9. Burkard, R.E., Offermann, J.: Entwurf von Schreibmaschinentastaturen mittels Quadratischer Zuordnungsprobleme. Zeitschrift fuer Operations Research 21 (1977) 121–132
10. Burkard, R.E.: Locations with Spatial Interactions: The Quadratic Assignment Problem. In: Mirchandani, P.B., Francis, R.L. (eds.): Discrete Location Theory. Wiley, New York (1991) 387–437

11. Burkard, R.E., Çela, E., Pardalos, P.M., Pitsoulis, L.: The Quadratic Assignment Problem. In: Du, D.Z., Pardalos, P.M. (eds.): Handbook of Combinatorial Optimization, Vol.3. Kluwer (1998) 241–337
12. Sahni, S., Gonzalez, T.: P-complete Approximation Problems. J. of ACM 23 (1976) 555–565
13. Gambardella, L.M., Taillard, E., Dorigo, M.: Ant Colonies for the Quadratic Assignment Problems. J. of the Operational Research Society 50 (1999) 167–176
14. Stuetzle, T., Dorigo, M.: ACO Algorithms for the Quadratic Assignment Problem. In: Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill (1999) 33–50
15. Ahuja, R.K., Orlin, J.B., Tiwari, A.: A Greedy Genetic Algorithm for the Quadratic Assignment Problem. Computers & Operations Research 27 (2000) 917–934
16. Fleurent, C., Ferland, J.A.: Genetic Hybrids for the Quadratic Assignment Problem. In: Pardalos, P.M., Wolkowicz, H. (eds.): Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol.16. AMS, Providence (1994) 173–188
17. Merz, P., Freisleben, B.: A Genetic Local Search Approach to the Quadratic Assignment Problem. Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97) (1997)
18. Li, Y., Pardalos, P.M., Resende, M.G.C.: A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. In: Pardalos, P.M., Wolkowicz, H. (eds.): Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol.16. AMS, Providence (1994) 237–261
19. Stuetzle, T.: Iterated Local Search for the Quadratic Assignment Problem. Tech. Report AIDA-99-03, Darmstadt, Germany (1999)
20. Boelte, A., Thonemann, U.W.: Optimizing Simulated Annealing Schedules with Genetic Programming. European J. of Operational Research 92 (1996) 402–416
21. Connolly, D.T.: An Improved Annealing Scheme for the QAP. European J. of Operational Research 46 (1990) 93–100
22. Battiti, R., Tecchiolli, G.: The Reactive Tabu Search. ORSA J. on Computing 6 (1994) 126–140
23. Skorin-Kapov, J.: Tabu Search Applied to the Quadratic Assignment Problem. ORSA J. on Computing 2 (1990) 33–45
24. Taillard, E.: Robust Taboo Search for the QAP. Parallel Computing 17 (1991) 443-455
25. Çela, E.: The Quadratic Assignment Problem: Theory and Algorithms. Kluwer, Dordrecht (1998)
26. Schrimpf, G., Schneider, K., Stamm-Wilbrandt, H., Dueck, V.: Record Breaking Optimization Results Using the Ruin and Recreate Principle. J. of Computational Physics 159 (2000) 139–171
27. Martin, O., Otto, S.W.: Combining Simulated Annealing with Local Search Heuristics. Annals of Operations Research 63 (1996) 57–75
28. Martin, O., Otto, S.W., Felten, E.W.: Large-step Markov Chains for the Traveling Salesman Problem. Complex Systems 5 (1991) 299–326
29. Mladenović, N., Hansen, P.: Variable Neighbourhood Search. Computers & Operations Research 24 (1997) 1097–1100
30. Burkard, R.E., Karisch, S., Rendl, F.: QAPLIB – A Quadratic Assignment Problem Library. J. of Global Optimization 10 (1997) 391–403
31. Taillard, E., Gambardella, L.M.: Adaptive Memories for the Quadratic Assignment Problem. Tech. Report IDSIA-87-97, Lugano, Switzerland (1997)