ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol. 35, No. 4

STAGNATION-PROTECTED TABU SEARCH VARIANTS FOR UNSTRUCTURED QUADRATIC ASSIGNMENT PROBLEMS❖

Alfonsas Misevičius, Arūnas Tomkevičius, Juozas Karbauskas

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

Abstract. Tabu search (TS) algorithms have been proven to be extremely efficient for solving combinatorial optimization problems. In this paper, we discuss new variants of the tabu search for the well-known combinatorial problem, the quadratic assignment problem (QAP). In particular, a so-called stagnation-protected tabu search (SPTS) strategy is proposed. The goal is to fight against the chaotic behaviour and stagnation phenomenon, especially at long runs of TS. SPTS seems to be quite useful for the unstructured (random) quadratic assignment problems. These problems, which resemble a "needle-in-a-haystack" problem, are hardly solvable by the ordinary heuristic algorithms and still remain a challenge for the QAP community. The results obtained from the experiments with SPTS on the unstructured QAPs taken from the QAP library QAPLIB demonstrate that this new strategy is superior to other tabu search algorithms.

Keywords: combinatorial optimization, quadratic assignment problem, heuristics, tabu search, stagnation-protected tabu search.

Introduction

The quadratic assignment problem (QAP) was introduced by Koopmans and Beckmann [11] as a mathematical model for the location of economic activities. Recently, this problem is frequently used as a "platform" for investigation of the performance of both exact and heuristic algorithms. The QAP can be formulated as follows. Let two matrices $A = (a_{ij})_{n\times n}$ and $B = (b_{kl})_{n\times n}$ be given. The goal is to find a permutation π of n elements that minimizes the following objective function:

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

where $\pi \in \Pi$, Π is the set of all possible permutations of n elements. A neighbourhood function N: $\Pi \to 2^{\Pi}$ assigns for each $\pi \in \Pi$ a set $N(\pi) \subseteq \Pi$ – the set of neighbouring solutions of π. An example of the neighbourhood function for the QAP is the 2-exchange function N_2. In this case, $N_2(\pi) = \{\pi' \mid \pi' \in \Pi, \rho(\pi, \pi') \le 2\}$, where $\pi \in \Pi$, and $\rho(\pi, \pi')$ is a distance between permutations π and π': $\rho(\pi,\pi') = \left|\{i \mid \pi(i) \ne \pi'(i)\}\right|$. Every neighbouring solution from $N_2(\pi)$ can be reached from the current solution π by a perturbation (move) *perturb*(π, i, j): $\Pi \times \mathrm{N} \times \mathrm{N} \to \Pi$, which gives π' such that $\pi'(i) = \pi(j)$, $\pi'(j) = \pi(i)$. We will also use the notation p_{ij} such that the expression $\pi' = \pi \oplus p_{ij}$ would mean that π' is obtained from π by applying *perturb*(π, i, j). The solution $\pi \in \Pi$ is said to be a locally optimal solution with respect to the neighbourhood N if $z(\pi') \ge z(\pi)$ for every $\pi' \in N(\pi)$. The solution π_{opt} is called an optimal solution if $\pi_{\text{opt}} \in \Pi_{\text{opt}} = \left\{\pi^{\nabla} \mid \pi^{\nabla} = \arg\min_{\pi\in\Pi} z(\pi)\right\}$.

The quadratic assignment problem is an NP-hard combinatorial optimization problem. It can be solved exactly for very small sizes only ($n \le 36$). Therefore, heuristic methods are extensively used for solving medium- and large-scale QAPs [2,4,15]. Tabu search (TS) algorithms, which are based on the intelligent neighbourhood search with memory, are among those that have been proven to be extremely efficient for the QAP [1,5,6,13,16,17]. Despite of this, there is still a room for further improvements of TS, especially if we are handling the unstructured (random) quadratic assignment problems (see, for example, the instances tai20a, tai25a, tai30a, tai35a, tai40a, tai50a, tai60a, tai80a, and tai100a taken from the QAP instances library QAPLIB [3]). These problems are characterized by random, uniformly distributed (regular) values of the data matrices. Although the data are regular, there is no regularity in the solution space. The landscapes of such problems are obviously disordered

❖ This work is supported by Lithuanian State Science and Studies Foundation through grant number T-06276.

with the enormous number of local optima. This is why the tabu search algorithms (as well as other heuristic algorithms) face severe difficulties when dealing with this class of problems. In fact, many heuristics work quite well with respect to the average quality of solutions; however, things look to be very pessimistic if one seeks for the pseudo-optimal (or near-pseudo-optimal) solutions. It seems that the large unstructured QAP instances ($n \geq 80$) are not practically solvable even to pseudo-optimality. The current work aims to focus on this issue.

The paper is organized as follows. In Section 1, we discuss our motivation of using the stagnation-protected tabu search strategy. Some variants of the stagnation-protected tabu search for the QAP are described in Section 2. In Section 3, we present the results of the computational experiments on the unstructured QAPs. Section 4 completes the paper with concluding remarks.

1. Stagnation-protected tabu search: preliminaries and motivation

For the unstructured problems, it is a common case that the search process converges quite rapidly, but without finding an optimal or even near-optimal solution. New (better) locally optimal solutions are encountered very easily at the early stage of the search process; however, as the search progresses, new better solutions grow rarer with each search iteration. It takes longer and longer time to reveal new record-breaking solutions at the later phases of the search. This tendency is particularly evident for extensive runs of time-consuming iterative heuristic algorithms like simulated annealing, tabu search, iterated local search, and others. For these methods, it can be observed that the time intervals between successive detections of a new better solution increase catastrophically without getting satisfactory results. This is even more true for large-scale problems. We call the above phenomenon the stagnation phenomenon (or simply stagnation). The following are the main reasons of stagnation:

- a huge number of locally optimal solutions over the solution space;
- many isolated local optima;
- complex, non-monotonic landscapes with small basins of attraction;
- cycles of the search trajectories;
- deterministic chaos (chaotic attractors).

Note that the large number of local optima does not necessarily imply complex landscape. There exist some special kind landscapes, which are relatively easy for heuristics (see Figure 1). Regarding deterministic chaos, it can be identified by the situation where getting stuck at local optima and cycling trajectories are absent – this is just the case of the standard (simple) tabu search – but the search configurations are still confined in limited parts of the search space. If these parts do not contain the pseudo-optimal solution, it may not be discovered.

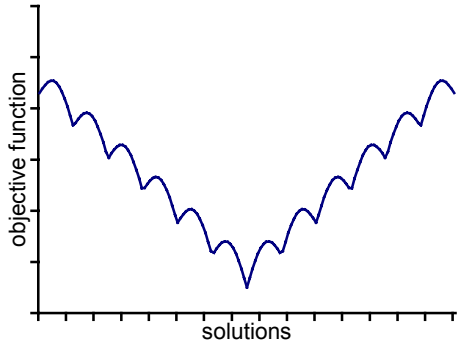

Figure 1. A special type landscape:"big valley"

The illustrations of stagnation episodes within the tabu search are shown in Figures 2–3. In Figure 2, the graph is presented which clearly demonstrates that the tabu search process returns to the same point (solution) from time to time. In Figure 3, the search trajectories are visualized by plotting the lines between the points that correspond to the solutions of the QAP (i.e. permutations). The co-ordinates (u, v) of these points are derived by using simple formulas: $u = (w \text{ div } GS) * SF$, $v = (w \bmod GS) * SF$, where $w = \left(\sum_{i=1}^{n} (i * \pi(i)) \right) \bmod H$, GS is a grid size, SF corresponds to a scaling factor (we used $GS = SF = 24$), and H denotes a hashing constant which is used by mapping a permutation π to a scalar w (in our experiments, $H = 10000$). It is obvious from Figure 3 that the search process is cyclic and chaotic.

Eliminating stagnation is one of the key issues by creating competitive TS algorithms. Several attempts to overcome the stagnant behaviour of TS have already been done (see, for example, the reactive tabu search [1], the tabu search with diversification strategies [10], the enhanced (iterated) tabu search [13]). The results from the experiments, however, demonstrate that finding the pseudo-optimal solutions of the unstructured problems remains a tough task even for the accurately designed and elaborated algorithms.

In this paper, we continue our endeavour to further enhance the performance of the tabu search algorithms. A new concept of the "stagnation-protected tabu search" (SPTS) is proposed. The focus is, in particular, on prevention of stagnation, minimizing cycling trajectories, avoiding becoming trapped at local optima, and, consequently, improving efficiency of the tabu search, especially at long and extra-long runs. Some different variants of SPTS for the QAP are described in the next section.

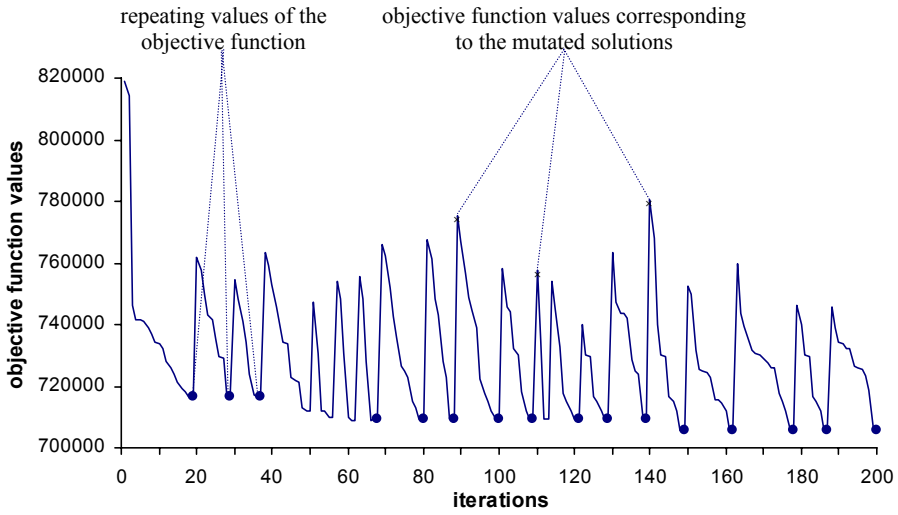

Figure 2. Illustration of the stagnation phenomenon for the unstructured QAP instance tai20a.
Note. The instance tai20a is from the library of the QAP instances QAPLIB [3]

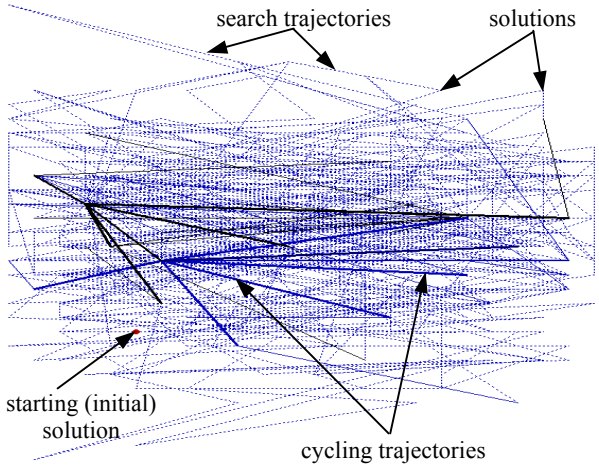

Figure 3. Visualization (hypothetical view) of solutions and search trajectories for the unstructured QAP instance tai20a

2. Stagnation-protected tabu search for the QAP

The paradigm we rely is the enhanced (iterated) tabu search[‡] [13], which, in turn, is based on the adapted robust tabu search (RoTS) algorithm [17] and the modified random pairwise interchange mutation procedure (see below). The goal of RoTS is to search for a better solution in the neighbourhood of the current solution, while mutation is responsible for escaping from the recent local optimum by generating diversified starting solutions. Combining of tabu search and mutation is done according to a so-called (Q, τ, 1)-strategy. In this case, the total number of SPTS iterations (global iterations) is equal to Q. At every global iteration, τ internal iterations are performed; in addition, one call to the mutation procedure takes place every τ iterations. The quantity $Q\tau$ stands for the overall number of iterations. The user can flexibly control the run time of the SPTS algorithm by choosing appropriate values of Q and τ.

At each internal iteration, the set of the neighbouring solutions of π – the set $N_2(\pi)$ – is considered and the move to the solution that improves most the objective function value is chosen. The complete evaluation of $N_2(\pi)$ takes $O(n^2)$ operations, except the first iteration, which takes $O(n^3)$ operations (see [17]). The tabu list (memory) is organized as an $n \times n$ integer matrix $T = (t_{ij})_{n\times n}$, where n is the problem size. At the beginning, all the entries of T are set to zero. As the search progresses, the entry t_{ij} stores the current iteration number plus the tabu tenure, h, i.e. the number of the future iteration starting at which ith and jth elements of the permutation may again be interchanged. The perturbation consisting of exchanging ith and jth elements is forbidden if the corresponding tabu criterion holds, i.e. t_{ij} is equal or greater than the current iteration number. However, the tabu status is ignored if an aspiration criterion is met, for example, the perturbation results in a solution that is better than the best so far (BSF) solution. Thus, the perturbation p_{ij} (i.e. the move from the current solution π to the solution $\pi \oplus p_{ij}$) is allowable if only it is not forbidden or aspired – an acceptance criterion is said to be satisfied.

Regarding mutation of solutions, we use the modified random pairwise interchange mutation (mrpi-mutation) procedure. It can simply be seen as a sequence of $\mu - 1$ random perturbations $p_{ri_1 ri_2}, p_{ri_2 ri_3}, \ldots, p_{ri_{\mu-1} ri_\mu}$, where ri is an array of random indices such that $1 \le ri_k \le n$, $k = 1, 2, \ldots, \mu$, $ri_k \ne ri_l$, $k = 1, 2, \ldots, \mu$, $l = 1, 2, \ldots, \mu$, $k \ne l$. The mutated permutation $\tilde{\pi}$ can thus be defined as a composition

‡ We assume that the reader is familiar with the basic principles of the TS method. Those interested in the fundamentals of TS are addressed to [7,8,9].

$\pi \oplus p_{ri_1ri_2} \oplus p_{ri_2ri_3} \oplus ... \oplus p_{ri_kri_{k+1}} \oplus ... \oplus p_{ri_{\mu-1}ri_\mu}$. This guarantees that $\rho(\pi, \pi^{\sim}) = \mu$, where π, $\pi^{\sim}$ are the actual and mutated permutations, respectively. The parameter μ ($1 < \mu \le n$) is called the mutation level. Note that the value of μ should be large enough to allow to leave the current local optimum and to move towards new regions in the solution space; on the other hand, μ should be small enough to keep characteristics of the good solutions since parts of these solutions may be close to the ones of the (pseudo-)optimal solution. In our implementation, μ is related to the problem size n, i.e. $\mu = \max\{2, \lfloor \xi \cdot n \rfloor\}$, where $\xi \in (0, 1]$ (we recommend $\xi = 0.3 \div 0.4$). It is easy to generate several distinct sequences of random perturbations and receive many mutated solutions instead of the single one. Our SPTS algorithm produces λ ($\lambda = n$) mutated solutions, but only the best "mutant" (i.e. the mutated solution that has the smallest objective function value) serves as an output of mrpi-mutation.

We are going to discuss the SPTS algorithm in more detail. So, let SPTS start with a random initial solution π°. Also, let q be the current global iteration number (counter) ($q = 1, 2, \ldots, Q$) and k be the current internal iteration number ($k = 1, 2, \ldots, \tau$). Furthermore, let denote the starting solution of the RoTS procedure by $\pi^{(0,q)}$ ($\pi^{(0,q)}$ ($q > 1$) corresponds to the solution which is the result of the mutation procedure performed at the $(q-1)$th iteration (note that $\pi^{(0,1)} = \pi^{\circ}$)). Then, the resulting solution of the kth iteration of the adapted RoTS procedure is the permutation $\pi^{(k,q)}$ such that:

$$\pi^{(k,q)} = \underset{\substack{i=1,\ldots,n-1,\ j=i+1,\ldots,n \\ acceptance\_criterion^{(i,j)} = \mathrm{TRUE}}}{\arg\min} z(\pi \oplus p_{ij}), \qquad (2)$$

where

$$acceptance\_criterion^{(i,j)} = \begin{cases} \mathrm{TRUE}, ((z < z_{\min}) \text{ and } (tabu\_criterion^{(i,j)} \\ \quad = \mathrm{FALSE})) \text{ or } (aspiration\_criterion^{(i,j)} = \mathrm{TRUE}) \\ \mathrm{FALSE}, \text{otherwise} \end{cases}, \qquad (3a)$$

$$tabu\_criterion^{(i,j)} = \begin{cases} \mathrm{TRUE}, (t_{ij} \ge k) \text{ and } (r \ge \alpha) \\ \mathrm{FALSE}, \text{otherwise} \end{cases}, \qquad (3b)$$

$$aspiration\_criterion^{(i,j)} = \begin{cases} \mathrm{TRUE}, ((z < z^{\bullet}) \text{ and} \\ \quad (tabu\_criterion^{(i,j)} = \mathrm{TRUE})) \text{ or } (z < z^{*}) \\ \mathrm{FALSE}, \text{otherwise} \end{cases}, \qquad (3c)$$

where i, j are the current indices of the permutation elements to be interchanged ($i = 1, \ldots, n-1, j = i+1, \ldots, n$); z denotes the current objective function value obtained by interchanging ith and jth elements of the actual permutation π,

$$z_{\min} = \begin{cases} \infty, i = 1 \text{ and } j = 2 \\ \min\limits_{\substack{\vee l,m: l,m \ge 1, m > l, (l-1)n - \frac{1}{2}l(l+1) + m < c \\ c = (i-1)n - \frac{1}{2}i(i+1) + j \\ acceptance\_criterion^{(l,m)} = \mathrm{TRUE}}} z(\pi \oplus p_{lm}), \text{otherwise} \end{cases};$$

r is a pseudo-random number from the interval [0, 1]; α plays the role of the randomization parameter (in fact, α is a probability that the tabu status will be ignored even if the aspiration criterion does not hold); finally, $z^{\bullet}$ denotes the intermediate BSF objective function value, i.e. $z^{\bullet} = \min\limits_{k'=0,\ldots,k-1} z(\pi^{(k',q)})$, and z^{*} is the overall BSF objective function value, i.e.

$$z^{*} = \begin{cases} z^{\bullet}, q = 1 \\ \min\left\langle \min\limits_{k'=0,\ldots,k-1} z(\pi^{(k',q)}), \min\limits_{q'=1,\ldots,q-1} \left[\min\limits_{k'=0,\ldots,\tau} z(\pi^{(k',q')}) \right] \right\rangle, \text{otherwise} \end{cases}.$$

The solution $\pi^{*} = \arg\min\limits_{q=1,\ldots,Q} \left[\arg\min\limits_{k=0,\ldots,\tau} z(\pi^{(k,q)}) \right]$ is regarded as the final solution of the SPTS procedure.

We call the tabu search variant defined by formulas (3a)–(3c) as basic. It will be referred to as SPTS1. In this variant, the only instrument for avoiding stagnation is the mutation procedure, which aims mainly at diversifying the search trajectories. The cycles still occur – we can make sure of this fact by looking at Figure 2.

In the next SPTS variants we propose, the stagnation avoidance is due to the extended tabu, aspiration, and/or acceptance criteria. These variants are denoted by SPTS2, SPTS3, SPTS4, and SPTS5. They are formally defined as follows:

SPTS2:

$acceptance\_criterion^{(i,j)}$ is equivalent to (3a)

s.t. (3b) and

$$aspiration\_criterion^{(i,j)} = \begin{cases} \mathrm{TRUE}, ((z < z^{\bullet}) \text{ and } (tabu\_criterion^{(i,j)} = ; \\ \quad \mathrm{TRUE}) \text{ and } (k > \eta)) \text{ or } (z < z^{*}) \\ \mathrm{FALSE}, \text{otherwise} \end{cases} \qquad (4)$$

SPTS3:

$acceptance\_criterion^{(i,j)}$ is equivalent to (3a)

s.t. (4) and

$$tabu\_criterion^{(i,j)} = \begin{cases} \mathrm{TRUE}, ((t_{ij} \ge k) \text{ and } (r \ge \alpha)) \text{ or } (z = z^{\blacktriangledown}) \\ \mathrm{FALSE}, \text{otherwise} \end{cases}, \qquad (5)$$

where $z^{\blacktriangledown}$ is the objective function value that corresponds to the previous local optimum, i.e. $z^{\blacktriangledown} = \begin{cases} \infty, q = 1 \\ \min\limits_{k'=0,\ldots,\tau} z(\pi^{(k',q-1)}), \text{otherwise} \end{cases}$, here q is the current global iteration number;

SPTS4:

$acceptance\_criterion^{(i,j)}$ is equivalent to (3a)

s.t. (3b) and

$$aspiration\_criterion^{(i,j)} = \begin{cases} \text{TRUE}, ((z < z^{\bullet}) \text{ and } (tabu\_criterion^{(i,j)} = \\ \quad \text{TRUE}) \text{ and } (k > \eta) \text{ and } (z \neq z^{\blacktriangledown})) \text{ or } (z < z^{*}) \\ \text{FALSE}, \text{otherwise} \end{cases}; \quad (6)$$

$$acceptance\_criterion^{(i,j)} = \begin{cases} \text{TRUE}, ((z < z_{\min}) \text{ and } (tabu\_criterion^{(i,j)} = \\ \quad \text{FALSE}) \text{ and } (z \neq z^{\blacktriangledown})) \text{ or} \\ \quad (aspiration\_criterion^{(i,j)} = \text{TRUE}) \\ \text{FALSE}, \text{otherwise} \end{cases}, \quad (7)$$

SPTS5:

s.t. (3b) and (4).

```
procedure StagnationProtectedTabuSearch;
  // input: π° – the initial solution, n – the problem size, Q, τ – the numbers of iterations, h – the tabu tenure, //
  //        α, β, γ, λ, ξ – the control parameters (α ∈ [0, 1], β > 0, γ > 0, λ ≥ 1, ξ ∈ (0, 1]) //
  // external variable: trace – the trail of the pairwise interchanges performed in the mutation procedure //
  // output: π* – the best solution found //
begin
  delay_interval := max(1, ⌊β·n⌋); intensification_interval := max(1, ⌊γ·h⌋); μ := max(2, ⌊ξ·n⌋); η := ⌊0.5·μ⌋;
  π := π°; π* := π°; z* := z(π*); T := 0;
  for q := 1 to Q do begin // outer cycle //
    if q = 1 then z▼ := ∞ else z▼ := z•; π• := π; z• := z(π•); i := 1; j := 1; k' := 1;
    calculate differences in the objective function values d_lm, l=1,…,n–1, m=l+1,…,n;
    for k := 1 to τ do begin // inner cycle //
      z_ini := z(π); z_min := ∞;
      repeat // target analysis: exploration of the neighbourhood N2(π) //
        i := IF(j < n, i, IF(i < n – 1, i + 1, 1)); j := IF(j < n, j + 1, i + 1);
        z := z_ini + d_ij;
        if acceptance_criterion^(i,j) = TRUE then begin z_min := z; i_min := i; j_min := j end
      until i = n – 1; // cycle continues until all the pairs (i, j) are examined //
      if z_min < ∞ then begin
        π := π ⊕ p_(i_min, j_min); // the current permutation is replaced by the new one //
        update differences d_lm, l = 1,…,n – 1, m = l + 1,…,n;
        if k > delay_interval then t_(i_min, j_min) := k + h else t_(i_min, j_min) := k + 1 //the move p_(i_min, j_min) becomes tabu //
      end; // if //
      if k – k' ≥ intensification_interval then begin
        switch to the alternative intensification (steepest descent);
        k' := k
      end; // if //
      if z(π) < z(π•) then begin π• := π; z• := z(π•) end; // saving the intermediate BSF solution //
      if z(π) < z(π*) then begin
        π* := π; z* := z(π*); // saving the overall BSF solution //
        T := 0 // wiping out of the tabu list to intensify the search in the neighbourhood of π* //
      end
    end; // for k... //
    if q < Q then begin
      T := 0; // emptying the tabu list //
      π := ModifiedRandomPairwiseInterchangeMutation(π•, n, λ, μ); // escaping from a local optimum //
      // π serves as a starting point for the next iteration //
      for k := μ – 1 downto μ – η do
        t_(trace_k, trace_(k+1)) := h // update the tabu list T taking into account the interchanged elements of π //
                                      // this step is optional //
    end // if //
  end // for q... //
end.
```

Figure 4. Template of the stagnation-protected tabu search algorithm for the QAP.
Note. For the detailed description of *acceptance_criterion*$^{(i,j)}$, see formulas (3a)–(3c), (4)–(7)

In SPTS1–5, we wipe out the tabu list T every τ iterations. In SPTS2–5, we additionally include η moves into the tabu list each time the mutation procedure is performed. Shortly speaking, we simply follow (in the reverse order) the trail of the random indices (i.e. the pairwise interchanges made in mrpi-mutation). The value of η is proportional to the actual mutation level, μ (we used $\eta = \lfloor 0.5\mu \rfloor$). Note that the aspiration criterion must be ignored as long as $k \le \eta$, unless $z < z^*$. Doing so prevents the TS algorithm from considering the elements that have recently been affected in the mutation procedure.

In SPTS3–5, the additional conditions "$z = z^{\blacktriangledown}$" and "$z \ne z^{\blacktriangledown}$" help preventing stagnation, in particular, by hindering the algorithm from falling back into the previous local optimum and long-term cycling.

```
function ModifiedRandomPairwiseInterchangeMutation(π, n, λ, μ);
  // input: π – the current solution (permutation), n – the problem size,  //
  //        λ – # of mutation iterations,  μ – the mutation level (strength)  //
  // external variable:  trace – the trail of the random pairwise interchanges  //
  // output: π~~ – the best mutated solution  //
begin
  π~ := π;  z̆ := ∞;
  for k := 1 to λ do begin
    // generation of array of random indices ri  //
    for i := 1 to n do ri_i := i;
    for i := 1 to μ do begin
      generate j, randomly, uniformly, i ≤ j ≤ n;
      ri := ri ⊕ p_ij
    end;
    // random pairwise interchange mutation  //
    for i := 1 to μ – 1 do  π~ := π~ ⊕ p_{ri_i, ri_{i+1}} ;
    if z(π~) < z̆ then begin
      π~~ := π~  // saving the best mutated solution with respect to the objective function //
      trace := ri  // saving the actual trail of the random pairwise interchanges  //
    end  // if  //
  end;  // for  //
  return π~~
end.
```

Figure 5. Template of the procedure for the modified random pairwise interchange mutation

1	**8**	**6**	**2**	**4**	**5**	**3**	**7**	**9**	current permutation π
6	**7**	**1**	**4**						random indices (array *ri*)
1	8	6	2	4	**3**	**5**	7	9	$\pi \oplus p_{67}$
5	8	6	2	4	**3**	**1**	7	9	$\pi \oplus p_{67} \oplus p_{71}$
2	8	6	**5**	4	**3**	**1**	7	9	$\pi \oplus p_{67} \oplus p_{71} \oplus p_{14}$
2	**8**	**6**	**5**	**4**	**3**	**1**	**7**	**9**	mutated permutation $\pi^{\sim}$

Figure 6. Example of the modified random pairwise interchange mutation ($\mu = 4$, $\rho(\pi, \pi^{\sim}) = 4$)

The detailed template of the stagnation-protected tabu search algorithm is presented in Figure 4, while the template of the mrpi-mutation procedure is given in Figure 5. In addition, an example of mrpi-mutation is shown in Figure 6.

The values of the control parameters are tuned to the maximum performance of the SPTS algorithm; in particular, Q is equal to 100 for shorter runs and Q is equal to 200 for longer runs; $\tau = n^2$; h is equal to $\lfloor 0.3n \rfloor$ for smaller instances and h is equal to $\lfloor 0.15n \rfloor$ for larger instances; $\alpha = 0.05$; $\beta = 1$; $\gamma = 2$; $\lambda = n$; finally, ξ is equal to 0.4 for smaller instances and ξ is equal to 0.3 for larger instances.

3. Computational experiments

In this section, we present the results of comparison of different algorithms. In the experiments, we handled only the unstructured (random) instances taken from the QAP library QAPLIB [3]. These instances are denoted by tai20a, tai25a, tai30a, tai35a, tai40a, tai50a, tai60a, tai80a, and tai100a (the numeral in the instance name denotes the size of the problem).

The algorithms used in the experiments are as follows: five variants of SPTS (SPTS1, SPTS2, SPTS3, SPTS4, SPTS5), as well as the robust tabu search algorithm (RoTS) [17], the reactive tabu search algorithm (ReTS) [1], the fast ant algorithm ("fast ant system" – FANT) [18], and the improved hybrid

genetic algorithm (IHGA) [12]. The following are the performance measures for the algorithms: a) the average deviation from the best known solution (BKS) – $\bar{\delta}$ ($\bar{\delta} = 100(\bar{z} - z_{\text{bks}})/z_{\text{bks}}$ [%], where $\bar{z}$ is the average objective function value over 10 restarts and z_{bks} is the objective function value that corresponds to the best known solution; b) the number of solutions that are within 1%-optimality (over 10 restarts) – $C_{1\%}$; c) the number of the best known solutions found – C_{bks}.

The experiments were organized in such a way that all the algorithms require approximately the same CPU time.

The results of the comparison are presented in Tables 1–2.

Table 1. Results of comparison of the SPTS variants for the unstructured QAP instances. The best results obtained are printed in bold face. CPU times per restart are given in seconds. 1.4 GHz PENTIUM computer was used in the experiments

Instance	n	BKV	$\bar{\delta}$, $C_{1\%}/C_{\text{bks}}$					CPU time
			SPTS1	SPTS2	SPTS3	SPTS4	SPTS5	
tai20a	20	703482 [a]	0.062 10/6	**0**	**0**	**0**	**0**	*2.5*
tai25a	25	1167256 [a]	0.106 10/6	**0**	0.043 10/8	0.051 10/7	**0**	*6.0*
tai30a	30	1818146 [a]	**0**	**0**	**0**	**0**	**0**	*12.4*
tai35a	35	2422002 [a]	0.138 10/5	0.026 10/8	0.170 10/4	0.090 10/7	**0**	*24.8*
tai40a	40	3139370 [a]	0.379 10/0	0.337 10/0	0.370 10/0	0.295 10/0	**0.256** 10/**1**	*43*
tai50a	50	4941410 [a]	0.525 10/0	0.462 10/0	0.555 10/0	0.495 10/**1**	**0.376** 10/0	*114*
tai60a	60	7205962 [b]	0.521 10/0	0.451 10/0	0.501 10/0	0.426 10/0	**0.374** 10/0	*247*
tai80a	80	13526696 [c]	0.379 10/0	0.395 10/0	**0.360** 10/0	0.428 10/0	0.382 10/0	*730*
tai100a	100	21071558 [c]	0.343 10/0	0.299 10/0	0.366 10/0	**0.278** 10/0	0.279 10/0	*2150*

[a] comes from [3]; [b] comes from [13]; [c] comes from [14].

Table 2. Results of comparison of SPTS5 with other algorithms for the unstructured QAP instances. The best results obtained are printed in bold face. CPU times per restart are given in seconds. 1.4 GHz PENTIUM computer was used in the experiments

Instance	n	BKV	$\bar{\delta}$, $C_{1\%}/C_{\text{bks}}$					CPU time
			RoTS	ReTS	FANT	IHGA	SPTS5	
tai20a	20	703482 [a]	0.052 10/7	0.093 10/8	0.574 6/0	0.003 10/9	**0**	*5.1*
tai25a	25	1167256 [a]	**0**	**0**	1.088 2/0	0.009 10/9	**0**	*12.4*
tai30a	30	1818146 [a]	0.047 10/6	**0**	0.821 6/0	**0**	**0**	*25.3*
tai35a	35	2422002 [a]	0.131 10/3	0.171 10/5	1.211 1/0	0.015 10/8	**0**	*51*
tai40a	40	3139370 [a]	0.543 10/0	0.239 10/0	1.036 1/0	0.309 10/0	**0.199** 10/**1**	*88*
tai50a	50	4941410 [a]	0.848 7/0	0.425 10/0	1.398 0/0	0.554 10/**1**	**0.291** 10/**1**	*231*
tai60a	60	7205962 [b]	0.802 8/0	0.518 9/0	1.304 0/0	0.549 10/0	**0.305** 10/0	*498*
tai80a	80	13526696 [c]	0.926 9/0	0.385 10/0	1.453 0/0	0.518 10/0	**0.297** 10/0	*1470*
tai100a	100	21071558 [c]	0.764 10/0	0.338 10/0	1.137 1/0	0.476 10/0	0.196 10/**1**♠	*4320*

♠ during long runs on the instance tai100a, SPTS5 was successful in finding the solution that is better than that reported in [14]; the new objective function value is equal to **21071540**.

It can be viewed from Tables 1–2 that our variants of the stagnation-protected tabu search yield very promising results for the unstructured QAP instances. For the most part, this is true for the variant SPTS5, which seems to be appreciably better than the remaining variants. The results from Table 2 also confirm that SPTS5 is superior to the robust and reactive tabu search, the improved hybrid genetic algorithm, and especially the fast ant algorithm. By the way, IHGA performs relatively well. This is very probably due to the fact that IHGA incorporates tabu search as a local improvement (post-crossover) procedure. On the other hand, SPTS5 is evidently better than the fast ant algorithm. The last one does not use tabu search. We conjecture that this circumstance is the main reason of poor results of FANT. On the whole, it could be stated that the tabu-search-based policy, in one or another form, is obligatory if one seeks for good solutions for the unstructured QAPs.

Finally, it should be emphasized that, in our experimentation, SPTS5 achieved the best-known solutions for all the unstructured QAP instances tested, except

the instances tai60a and tai80a. SPTS5 also found new record-breaking solution for the largest unstructured QAP instance from QAPLIB – tai100a.

4. Concluding remarks

The unstructured instances of the quadratic assignment problem still pose a real challenge for the designers of heuristic algorithms. New algorithms capable of the effective exploration of the unstructured solution spaces are needed. The stagnation-protected tabu search (SPTS) is along this line of thinking. SPTS aims, in particular, to improve the performance of the tabu search by fighting against the stagnation phenomenon, which is one of the main barriers of the straightforward TS algorithms, especially in the cases where long and extra-long runs are necessary.

In this paper, five variants of SPTS with the different extended tabu, aspiration, and acceptance criteria are proposed. The results obtained from the experiments with SPTS demonstrate that the situation with the unstructured QAP instances does not seem hopeless. Our SPTS algorithms produce quite encouraging solutions for these instances. For the unstructured QAPs, SPTS obviously outperforms both the earlier tabu search versions and other heuristic algorithms. In addition, the new best-known solution for the unstructured instance tai100a was discovered.

Two main directions for the future research are the further investigation of innovative tabu, aspiration, and acceptance criteria as well as trying other more advanced mutation strategies. Intelligent recombination of pairs (or several) solutions instead of crude random mutation might be experienced, too. It is also worthy incorporating SPTS into the other modern meta-heuristics like hybrid genetic or ant algorithms.

References

[1] **R. Battiti, G. Tecchiolli.** The reactive tabu search. *ORSA Journal on Computing*, 1994, *Vol*.6, 126-140.

[2] **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*, *Kluwer, Dordrecht*, 1998, *Vol*.3, 241–337.

[3] **R.E. Burkard, S. Karisch, F. Rendl.** QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 1997, *Vol*.10, 391–403. [*See also* `http://www.seas.upenn.edu/qaplib/.`]

[4] **E. Çela.** The Quadratic Assignment Problem: Theory and Algorithms. *Kluwer, Dordrecht*, 1998.

[5] **J. Chakrapani, J. Skorin-Kapov.** Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 1993, *Vol*.41, 327–341.

[6] **Z. Drezner.** The extended concentric tabu for the quadratic assignment problem. *European Journal of Operational Research*, 2005, *Vol*.160, 416–422.

[7] **M. Gendreau.** An introduction to tabu search. *In F.Glover, G.Kochenberger (eds.), Handbook of Metaheuristics*, *Norwell: Kluwer*, 2002, 37–54.

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

[9] **A. Hertz, E. Taillard, D. de Werra.** Tabu search. *In E. Aarts, J.K. Lenstra (eds.), Local Search in Combinatorial Optimization, Chichester: Wiley,* 1997, 121–136.

[10] **J.P. Kelly, M. Laguna, F. Glover.** A study of diversification strategies for the quadratic assignment problem. *Computers & Operations Research*, 1994, *Vol*.21, 885–893.

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

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

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

[14] **A. Misevičius, A. Lenkevičius, D. Rubliauskas.** Iterated tabu search: an improvement to standard tabu search. *Information Technology and Control*, 2006, *Vol*.35, 187–197.

[15] **P.M. Pardalos, F. Rendl, H. Wolkowicz.** The quadratic assignment problem: a survey and recent developments. *In P.M. Pardalos, H. Wolkowicz (eds.), Quadratic Assignment and Related Problems. DIMACS Series an Discrete Mathematics and Theoretical Computer Science*, *Vol*.16, Providence: AMS, 1994, 1–41.

[16] **J. Skorin-Kapov.** Extension of a tabu search adaptation to the quadratic assignment problem. *Computers & Operations Research*, 1994, *Vol*.21, 855-865.

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

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

Received October 2006.