Mathematical Modelling and Analysis
Volume 9 Number 2, 2004, pages 149–168
© 2004 Technika ISSN 1392-6292

AN IMPROVED HYBRID OPTIMIZATION ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM

A. MISEVIČIUS

Kaunas University of Technology, Department of Practical Informatics
Studentų St. 50–400a, 3031 Kaunas, Lithuania
E-mail: `alfonsas.misevicius@ktu.lt`

Received November 20, 2003; revised January 29, 2004

Abstract. In this paper, we present an improved hybrid optimization algorithm, which was applied to the hard combinatorial optimization problem, the quadratic assignment problem (QAP). This is an extended version of the earlier hybrid heuristic approach proposed by the author. The new algorithm is distinguished for the further exploitation of the idea of hybridization of the well-known efficient heuristic algorithms, namely, simulated annealing (SA) and tabu search (TS). The important feature of our algorithm is the so-called "cold restart mechanism", which is used in order to avoid a possible "stagnation" of the search. This strategy resulted in very good solutions obtained during simulations with a number of the QAP instances (test data). These solutions show that the proposed algorithm outperforms both the "pure" SA/TS algorithms and the earlier author's combined SA and TS algorithm.

Key words: hybrid optimization, simulated annealing, tabu search, quadratic assignment problem, simulation

1. Introduction

The quadratic assignment problem (QAP) is the famous combinatorial optimization problem. It is formulated as follows. Let two matrices $A = (a_{ij})_{n\times n}$ and $B = (b_{kl})_{n\times n}$ and the set Π of permutations of the integers from 1 to n be given. Find a permutation $\pi = \big(\pi(1), \pi(2), \ldots, \pi(n)\big) \in \Pi$ that minimizes

$$z(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}. \tag{1.1}$$

One of the important applications of the QAP is computer-aided design (CAD), namely, the placement of electronic components [23, 26, 40]. In this context, the entries of the matrix $A = (a_{ij})_{n\times n}$ can be interpreted as the numbers of connections (nets) between components. The entries of the matrix $B = (b_{kl})_{n\times n}$ represent distances between locations (positions). The permutation $\pi = \big(\pi(1), \pi(2), \ldots, \pi(n)\big)$ corresponds to a certain placement of components to locations ($\pi(i)$ denotes the location that component i is placed into). Thus, z, or more precisely $\frac{1}{2}z$, can be treated as an estimation of total wire length obtained when n components are placed into n locations (see Fig. 1). Description of the other applications of the QAP one can be found in [7, 8, 10].

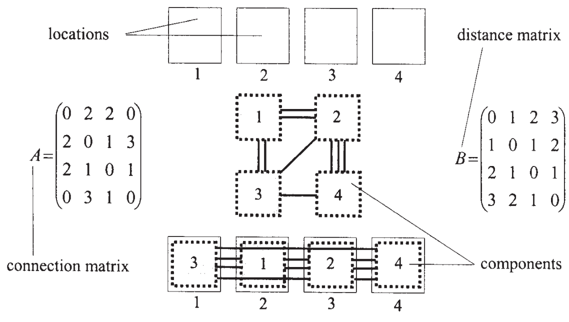
locations
distance matrix
1
2
3
4
$A=\begin{pmatrix}0&2&2&0\\2&0&1&3\\2&1&0&1\\0&3&1&0\end{pmatrix}$
$B=\begin{pmatrix}0&1&2&3\\1&0&1&2\\2&1&0&1\\3&2&1&0\end{pmatrix}$
connection matrix
components

Figure 1. Graphical interpretation of the quadratic assignment problem. For given matrices A and B the permutation corresponding to optimal assignment is as follows:(2,3,1,4). The connection length that corresponds this assignment is equal to 12.

It has been proved that the QAP (like many other combinatorial optimization problems) is NP-hard [38]. For example, QAPs of size $n > 36$ are not, to this date, practically solvable in terms of obtaining exact solutions. Therefore, heuristic techniques have to be used for solving medium- and large-scale QAPs (see, for example, [14, 15, 16, 29, 32, 41]; for a more detailed list of heuristics for the QAP, see [8, 10, 37]).

First we introduce some basic definitions related to the combinatorial (discrete) optimization. So, let S be a set of solutions of a combinatorial optimization problem with an objective function $f : S \to R^1$ (without loss of generality, we assume that f seeks a global minimum). 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 neighbouring solution $s' \in N(s)$ can be reached directly from the current solution s by an operation, which is called a *move*. Usually, the move follows the objective function evaluation which is called a trial. An iteration is said to be performed when $|N(s)|$ trials are done.

Regarding the QAP, $\Pi = \{\pi | \pi = (\pi(1), \pi(2), \ldots, \pi(n))\}$, where $|\Pi| = n!$, corresponds to S, and z (defined according to (1.1)) plays a role of the objective function. In the case of the QAP, the commonly used neighbourhood function is so-called 2-exchange (pairwise exchange) function N_2 which can be defined as follows:

$$N_2(\pi) = \{\pi' | \pi' \in \Pi,\ \rho(\pi, \pi') = 2\}, \tag{1.2}$$

where $\pi \in \Pi$ and $\rho(\pi, \pi')$ is a "distance" between the current permutation π and the neighbouring one π' :

$$\rho(\pi, \pi') = \sum_{i=1}^{n} sgn|\pi(i) - \pi'(i)|\,.$$

In this case, a move from the permutation π to the permutation π' can formally be defined by using a 2-way perturbation operator

$$p_{ij} : \quad \Pi \to \Pi \ \big(i, j = 1, 2, \ldots, n;\ i \neq j\big)\,,$$

which exchanges ith and jth elements in the current permutation (notation $\pi' = \pi \oplus p_{ij}$ means that π' is obtained from π by applying the perturbation p_{ij}). For a permutation π and a perturbation p_{ij}, it is more efficient to compute $\varDelta z(\pi, i, j) = z(\pi \oplus p_{ij}) - z(\pi)$ than $z(\pi \oplus p_{ij})$: the direct computation of $z(\pi \oplus p_{ij})$ needs time $O(n^2)$, whereas $\varDelta z(\pi, i, j)$ can be calculated in $O(n)$ operations:

$$\begin{aligned}\varDelta z\,(\pi, i, j) = (a_{ij} - a_{ji})\big(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)}\big) + \sum_{k=1, k\neq i,j} \big[(a_{ik} - a_{jk})\\ \times \big(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}\big) + (a_{ki} - a_{kj})\big(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)}\big)\big], \quad (1.3)\end{aligned}$$

where $a_{ii}(b_{ii}) = const,\ i = 1, 2, \ldots, n$. Moreover, for two consecutive permutations π and $\pi' = \pi \oplus p_{uv}$, if all the values $\varDelta z(\pi, i, j)$ have been stored (i.e. already calculated in previous iteration), then the values

$$\varDelta z(\pi', i, j) = z(\pi' \oplus p_{ij} - z(\pi'),\ i \neq u, v,\ j \neq (u, v)$$

can be computed in time $O(1)$ [42]:

$$\begin{aligned}\varDelta z\,(\pi', i, j) = \varDelta z\,(\pi, i, j) + (a_{iu} - a_{iv} + a_{jv} - a_{ju})(b_{\pi(i)\pi(u)}\\ - b_{\pi(i)\pi(v)} + b_{\pi(j)\pi(v)} - b_{\pi(j)\pi(u)}) + (a_{ui} - a_{vi} + a_{vj} - a_{uj})\\ (b_{\pi(u)\pi(i)} - b_{\pi(v)\pi(i)} + b_{\pi(v)\pi(j)} - b_{\pi(u)\pi(j)})\,. \quad (1.4)\end{aligned}$$

However, if $i = u$ or $i = v$ or $j = u$ or $j = v$, then the formula (1.3) should be applied.

Two main alternatives exist when exploring the neighbouring solutions. First, choose the next potential solution at random. Second, explore the neighbourhood in a systematic way. In the case of the 2-exchange neighbourhood function N_2, the order of search can be established by a sequence $\{p_{i^{(k)}j^{(k)}}\}$. The indices $i^{(k)}, j^{(k)}$ are easily determined by the following expression

$$\begin{cases} i^{(k)} = iif(j^{(k-1)} < n, i^{(k-1)},\ iif(j^{(k-1)} < n-1, i^{(k-1)} + 1, 1)),\\ j^{(k)} = iif(j^{(k-1)} < n,\ j^{(k-1)} + 1, i^{(k)} + 1),\end{cases}$$

where

$$iif(x, y_1, y_2) = \begin{cases} y_1, & \text{if } x\ is\ true\\ y_2, & \text{otherwise;}\end{cases}$$

k is the current trial number $(k = 1, 2, \ldots)$; $i^{(k)}, j^{(k)}$ are new indices; $i^{(k-1)}, j^{(k-1)}$ are old indices $(i^{(0)} = 1, j^{(0)} = 1), K = |N_2| = n(n-1)/2$ trials are needed in order to explore all the solutions of N_2.

The remaining part of this paper is organized as follows. In Section 2 hybrid optimization strategies (paradigms) are outlined, whereas in Sections 3, 4 we survey the simulated annealing (SA) and tabu search (TS) techniques, which were used in our hybrid approach. Section 5 describes an improved hybrid optimization algorithm for the quadratic assignment problem. The results of simulations are presented in Section 6. Finally, Section 7 completes the paper with concluding remarks.

2. Hybrid optimization strategies

Over the last years, hybrid optimization algorithms have become very popular among researchers in combinatorial optimization. First of all, this is due to promising results obtained by using hybrid (combined) approaches. Combinations of both single-solution algorithms (such as greedy heuristic search, simulated annealing, tabu search) and population-based algorithms (such as genetic, evolutionary algorithms) have been proven to be extremely efficient for many combinatorial optimization problems [17, 19, 36]. Different hybrid meta-heuristics, i.e. paradigms of hybridization of heuristics can be proposed [43]. Further, two simple paradigms are outlined very roughly: first, a sequential hybridization, second, an embedded hybridization. Without loss of generality, we discuss the hybrid scheme that consists of two heuristics only.

```
procedure sequential_hybridization /* $H_1 + H_2 + \ldots$ */
   ...
   apply heuristic $H_1$;
   apply heuristic $H_2$;
   ...
end
```

Figure 2. Pseudo-code for the framework of the sequential hybrid meta-heuristic.

So, in the first case, the self-contained heuristics H_1 and H_2 are executed in a sequence (one after other), the heuristic H_2 using the output of the heuristic H_1 as its input (i.e. the heuristics act in a pipeline fashion). Here, H_2 can also be thought of as a "post-analysis" procedure which is applied to the solution found by H_1. For example, a greedy (or more sophisticated) heuristic can be used to generate good initial solutions for the genetic/evolutionary algorithm [19]. In the second variant, the heuristic H_2 is embedded into heuristic H_1 (i.e. heuristics act as cooperating agents). For example, deterministic local search technique may be embedded into simulated annealing (as proposed in [31]) or genetic algorithm (see, for example, [12, 17, 19]). For a more formal presentation of the above paradigms, see Figures 2,3.

```
procedure embedded_hybridization /*$H_1(H_2(\ldots))$*/
   ...
   apply heuristic $H_1$;
   ...
end
procedure $\mathbf{H_1}$
   ...
   apply heuristic $H_2$;
   ...
end
```

Figure 3. Pseudo-code for the framework of the embedded hybrid meta-heuristic.

However, disposing of these two hybridization paradigms only may be insufficient for complex combinatorial problems, like the quadratic assignment problem. These problems can be seen as highly "discontinuous": if one "walks" in a fictitious solution space, the qualities of the solutions can differ dramatically, i.e. the "landscapes" of these problems are very rugged. Another distinguishing feature is a presence of a big number of local optima, which are often spread over the whole solution space (see Figure 4).

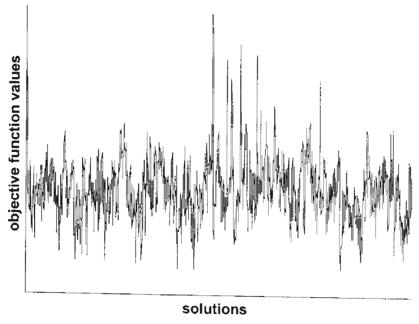

Figure 4. Example of a complex "landscape".

In these situations, the strategies described above usually face a phenomenon called a "stagnation" of the search (also known as a "chaotic attractor" [5]). This means that the search trajectory is confined in a limited part (region) of the solution space: if this part does not contain the global optimum, it will never be found.

An enhanced hybrid strategy (we call it an iterative hybridization) is designed in order to try to overcome these difficulties. In fact, this hybridization strategy is an extension of the sequential hybridization. The extension is constructed in such a manner that self-contained heuristics, say H_1 and H_2, are used in a cyclic (iterative) way, i.e. the heuristic H_2 uses the output of the heuristic H_1, and the heuristic H_1 uses the output of the heuristic H_2 (starting from the second iteration). The paradigm of the iterative hybrid strategy is shown in Fig. 5.

In our hybrid optimization algorithm for the QAP, which will be presented in Section 5, we use simulated annealing approach based procedure in the role of

```
procedure iterative_hybridization
   /*$(H_1 + H_2 + \cdots) + (H_1 + H_2 + \ldots) + \ldots$*/
   ...
   repeat
     apply heuristic $H_1$;
     apply heuristic $H_2$;
     ...
   until termination  criterion satisfied
end
```

Figure 5. Pseudo-code for the framework of the iterative hybrid meta-heuristic.

the heuristic H_1, and tabu search approach based procedure as the "post-analysis" heuristic H_2. Note that hybridization of SA and TS was also used for solving the other problems (see, for example, [18, 36, 45].) Before describing the details of our hybrid algorithm, we give short overviews of SA and TS approaches.

3. Simulated annealing

Simulated annealing originated in statistical mechanics. It is based on a Monte Carlo model that was used by Metropolis et al., 1953 [33], to simulate energy levels in cooling solids. Boltzmann's law was used to determine the probability of accepting a perturbation resulting in a change $\varDelta E$ in the energy at the current temperature t :

$$P = \begin{cases} 1, & \varDelta E < 0, \\ e^{-\varDelta E/Ct}, & \varDelta E \geq 0, \end{cases}$$

where C is a Boltzmann's constant. Cerny, 1982 [11], and Kirkpatrick et al., 1983 [27] have applied firstly SA to solve combinatorial optimization problems. Several researchers tested SA on the QAP, as well [6, 13, 35, 44]. The principle of the simulated annealing is simple: start from a random solution. Given a solution s select a neighbouring solution s' and compute the difference of the objective function values, $\varDelta f = f(s') - f(s)$. If the objective function value is improved $(\varDelta f < 0)$, then replace the current solution by the new one, i.e. perform a move, and use resulting configuration as a starting point for the next trial. If $\varDelta f \geq 0$, then accept a move with probability

$$P(\varDelta f) = e^{-\varDelta f/t}, \tag{3.1}$$

where t is the current temperature (Boltzmann's constant is not required when we apply the algorithm to combinatorial problems). Regarding the above probabilistic acceptance, it is achieved by generating a random number in [0,1] and comparing it against the threshold $e^{-\varDelta f/t}$ (here, the exponential function plays a role of an acceptance function). The procedure is repeated until a termination criterion is satisfied, for example, a predefined number of trials has been performed. As a resulting solution, usually the "best so far" (BSF) solution (instead of so-called "where you are" (WYA) solution) is returned by the algorithm. The paradigm of SA in a high-level algorithmic language form is presented in Fig. 6.

```
procedure simulated_annealing
   /* input: $s^{(0)}$ – the initial solution; output: $s^*$ – the best solution found */
   set $s = s^{(0)}$, $s^* = s$;
   determine the initial temperature $t_0$, set $t = t_0$;
   repeat /* main cycle */
     select new solution $s'$ from the neighbourhood
     of the current solution $s$;
     calculate $\Delta f = f(s') - f(s)$;
     generate uniform random number $r$ from the interval [0,1];
     if $(\Delta f < 0)$ or $(r < e^{-\Delta f/t})$ then set  $s = s'$;
     /* replace the current solution by the new one*/
     if $f(s) < f(s^*)$ then  set $s^* = s$;  /* save the best so far solution */
     update the temperature $t$;
   until stopping condition is satisfied
end
```

Figure 6. Pseudo-code for the simulated annealing.

SA algorithms differ mainly with respect to a cooling (annealing) schedule implemented. The cooling schedule, in turn, is specified by:

a) an initial (and final) value of the temperature,

b) an updating function for changing the temperature.

The most important thing is how the initial temperature t_0 is specified. If the initial value of the temperature is chosen too high, then too many bad uphill moves are accepted, while if it is too low, then the search will quickly drop into a local optimum without possibility to escape form it. Thus, an optimum initial temperature must be somewhere between these two extremes.

The temperature is not a constant, but changes over time according to the updating function. One of the popular updating functions (known as Lundy-Mees schedule) is characterized by the following relation [30]:

$$t_{k+1} = \frac{t_k}{1+\beta t_k}, \quad k = 0, 1, \ldots, \ t_0 = const, \ \beta << t_0 \,. \qquad (3.2)$$

It is easy to relate the coefficient β and the number of trials, i.e. the schedule length, L, under condition that the initial and final values of the temperature (t_0, t_f) are predefined:

$$\beta = \frac{t_0 - t_f}{L t_0 t_f} \,. \qquad (3.3)$$

In theory, the simulated annealing procedure should be continued until the final temperature t_f is zero, but in practice the other stopping criteria are applied, for example:

a) the value of the objective function has not decreased for a large number of consecutive trials;

b) the number of accepted moves has become less than a certain small threshold for a large number of consecutive trials;

c) a fixed a priori number of trials/iterations has been executed.

For more details about simulated annealing, the reader is referred to [1, 2, 28].

4. Tabu search

Tabu search technique was developed by Hansen and Jaumard, 1987 [24], and Glover, 1989, 1990 [20, 21]. TS has been proven to be a powerful tool for solving many combinatorial problems, among them the QAP (see, for example, [5, 14, 39, 42]). Tabu search, like simulated annealing, is based on the neighbourhood search with local-optima avoidance but in a rather deterministic way. The key idea of tabu search is allowing climbing moves when no improving neighbouring solution exists. However, some moves are to be forbidden at a present search iteration in order to avoid cycling.

TS starts from an initial solution s, maybe, randomly generated in S and moves repeatedly from a solution to a neighbouring one. At each step of the procedure, a set (subset) $N(s)$ of the neighbouring solutions of the current solution s is considered and the move that improves most the objective function value f is chosen. If there are no improving moves, TS algorithm chooses one that least degrades (increases) the objective function, i.e. a move is performed to the best neighbour s' in $N(s)$ (even if $f(s') > f(s)$).

In order to avoid returning to the local optimal solution just visited, the reverse move must be forbidden (prohibited). This is done by storing this move (or an attribute of the move) in a memory (or more precisely short-term-memory) managed like a circular list T and called a tabu list. The tabu list keeps information on the last h ($h = |T|$) moves which have been done during the search process (the parameter h is called a tabu list size). Thus, a move from s to s' is considered as tabu if it (or its attribute) is contained in the list T. This way of proceeding hinders the algorithm from returning to a solution reached in the last h iterations. However, it might be worth returning after a while to a solution visited previously to search in another direction. Consequently, an aspiration criterion is introduced to permit the tabu status to be dropped under certain favourable circumstances. Typically, a tabu move from s to s' is permitted if $f(s') < f(s^*)$, where s^* is the best solution found so far. The resulting decision rule within TS may thus be described as follows: replace the current solution s by the new solution s', if

$$f(s') < f(s^*) \quad \text{or} \quad (s' = arg \min_{s'' \in N(s)} f(s^{''}) \text{ and } s' \text{ is not tabu}). \tag{4.1}$$

The whole process is stopped as soon as a termination criterion is satisfied (for example, a fixed a priori number of trials has been performed). The tabu search paradigm is shown in Figure 7.

The TS algorithms differ mainly with respect to the basic ingredients discussed above (i.e. tabu list, aspiration criterion) and other additional features (for example, a long-term-memory, diversification mechanisms, etc.). The main forms of the tabu search are: deterministic tabu search (strict tabu search, fixed tabu search, reactive tabu search) and stochastic tabu search (probabilistic tabu search, robust tabu search). For more details on the TS technique, the reader is addressed to [22, 25].

```
procedure tabu_search
   /* input: $s^{(0)}$ – the initial solution; output: $s^*$ – the best solution found */
   set $s = s^{(0)}$, $s^* = s$;
   initialize the tabu list $T$;
   repeat /* main cycle*/
      given neighbourhood function $N$,
      tabu list $T$ and aspiration criterion,
      find the best possible solution $s' \in N(s)$;
      set $s = s'$; /* replace the current solution by the new one*/
      insert the solution $s$ (or its attribute)
      into the tabu list $T$;
      if $f(s) < f(s^*)$ then set $s^* = s$ /*save the best so far solution */
      update the tabu list $T$ (or its size) (if necessary)
   until stopping condition is satisfied
end
```

Figure 7. Pseudo-code for the tabu search.

5. An improved hybrid simulated annealing and tabu search algorithm for the QAP

Now we describe details of our hybrid strategy for the QAP. It is distinguished for the following structure: 1) simulated annealing algorithm, 2) tabu search algorithm, and 3) hybridization scheme.

5.1. Simulated annealing algorithm for the QAP (SA-QAP)

One of the important features of our implementation of the simulated annealing is that we use an extended approach of determining the values of the initial and final temperatures (these values are crucial for the SA algorithm, as mentioned in Section 3). Typically, the initial (and final) temperature is a function of the minimum and maximum differences in the objective function values obtained by performing a fixed number of moves before starting the annealing [13]. In our SA algorithm, we ignore the maximum difference; instead, we use the average difference. The formula of calculating the initial and final temperatures (t_0, t_f) looks, thus, as follows:

$$\begin{cases} t_0 = (1-\lambda_1)\Delta z_{min} + \lambda_1 \Delta z_{avg}, \\ t_f = (1-\lambda_2)\Delta z_{min} + \lambda_2 \Delta z_{avg}\,, \end{cases} \tag{5.1}$$

where $\Delta z_{min}, \Delta z_{avg}$ are, respectively, the minimum and average differences in the objective function values; $\lambda_1 \in (0,1]; \lambda_2 \in [0,1); \lambda_1 > \lambda_2$. In fact, the execution of the algorithm is controlled by operating with these factors. By choosing appropriate values of λ_1 and λ_2, one can control the cooling process flexibly. For example, having $\lambda_2 = const$ it is obvious that the larger the value of λ_1, the higher the initial temperature; on the other hand, the larger the difference $\lambda_1 - \lambda_2$, the more "rapid" the cooling. We use $\lambda_1 = 0.5$ and $\lambda_2 = 0.05$.

Another property of the SA algorithm is related to an intelligent annealing technique. The key idea is that the temperature is not monotone decreasing, but oscillating; that is, a re-annealing (a repeating sequence of coolings and heatings) is considered instead of the straightforward annealing (see also [3, 6]). We propose the re-annealing technique which is based on so-called dynamic cooling schedule. The parameters of this schedule (schedule length, initial and final temperatures) are adaptively changed during execution of the algorithm. We use a Lundy-Mees function based temperature oscillation (LM-oscillation) that is "process-dependent", i.e. it depends upon the former "behaviour" of the (re)annealing. The schedule is as follows: set the schedule length L to $Q_{SA}n(n-1)/2$ ($Q_{SA} \geq 1$) and start with the initial temperature defined by the formula (5.1). The temperature is then being updated according to the formula (3.2), the coefficient β is known from the formula (3.3). When $0.5|N_2| = n(n-1)/4$ consecutive moves are rejected, stop the (preliminary) cooling. After cooling is stopped, the temperature is immediately increased (i.e. the system is "heated up"), and the annealing with the new parameters starts. Additionally, a deterministic downhill search procedure CRAFT [4] is applied to the best solution found. The process is continued until a stopping criterion is satisfied, i.e. the current iteration number exceeds Q_{SA}, where Q_{SA} is the maximum number of iterations.

The detailed template of the SA algorithm for the QAP (SA-QAP) is presented in Figure 8 (see also [35]).

5.2. Tabu search algorithm for the QAP (TS-QAP)

Our version of the tabu search algorithm for the QAP is based on a slightly modified robust tabu search (RTS) procedure due to Taillard [42]. Very roughly, our algorithm consists of maintaining the tabu list T by constructing and updating it. The tabu list 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 number of the iteration plus the tabu list size, h, i.e. the number of the future iteration starting at which ith and jth elements of the permutation may again be interchanged. In this case, a move consisting of exchanging ith and jth elements is tabu if the value of t_{ij} is equal or greater than the current iteration number (this means that ith and jth elements were interchanged during the last h iterations).

The tabu list size h is not a constant – it is changed randomly during the search process. In our implementation, h is chosen between $h_{min} = 0.4n$ and $h_{max} = 0.6n$ and changed every $2h_{max}$ iterations. The standard aspiration criterion is used, i.e. the tabu status of a move is ignored (a tabu move is allowed to be selected) if the move results in a solution (permutation) that is better than the best one found so far.

In addition, we use the formula (1.4) to accelerate the evaluation of the neighbouring solutions of the current solution. Thus, the complete evaluation of the 2-exchange neighbourhood takes $O(n^2)$ operations, except the first iteration, which takes $O(n^3)$ operations (see formula (1.3). The run time of the tabu search procedure is controlled by the number of iterations, QTS.

The detailed template of the TS algorithm (TS-QAP) is presented in Figure 9.

```
procedure SA-QAP /* simulated annealing algorithm for the QAP */
    /* input: $\pi$ – the current (initial) permutation, $n$ – the problem size */
    /* $Q_{SA}$ – the number of iterations ($Q_{SA} \geq 1$) */
    /* $\lambda_1, \lambda_2$ – the initial and final temperature factors
    ($0 < \lambda_1 \leq 1, 0 \leq \lambda_2 < 1, \lambda_1 > \lambda_2$) */
    /* output: $\pi^*$ – the best permutation found */
    $\pi^* = \pi$;
    found $\Delta z_{min}, \Delta z_{avg}$ by performing $n(n-1)/2$ random moves
    starting from $\pi$;
    $M := Q_{SA}n(n-1)/2,\ L_0 := M$;
    /* $M$ – the number of trials, $L_0$ – the initial cooling schedule length */
    initialize cooling schedule parameters $L, t_0, t_f, \beta$;
    $t := t_0, i := 1, j := 1, rejected\_count := 0, oscillation :=' FALSE'$;
    for $k := 1$ to $M$ do begin /* main loop */
      $i := iif(j < n, i, iif(i, n-1, i+1, 1)),\ j := iif(j < n, j+1, i+1)$;
      calculate $\Delta = \Delta z(\pi, i, j)$;
      /* $\Delta z(\pi, i, j)$ is the current difference of the objective function values */
      if $\Delta < 0$ then $accept :=' TRUE'$
              else begin
                generate uniform random number $r$ from the interval [0,1];
                if $r < exp(-\Delta/t)$
                then $accept :=' TRUE'$ else $accept :=' FALSE'$
              end /* else */
      if $accept =' TRUE'$ then begin
              $\pi := \pi \oplus p_{ij}$; /* replace the current permutation by the new one */
              if $z(\pi) < z(\pi^*)$ then $\pi^* := \pi$;
               /* save the best permutation found so far */
              if $\Delta \neq 0$ then $rejected\_count := 0$
              end
              else $rejected\_count := rejected\_count + 1$;
      if ($rejected\_count \geq n(n-1)/4$) or ($t$ is at lowest point)
              then begin
              if $oscillation =' FALSE'$
              then begin $L^* := k, t^* := t, oscillation :=' TRUE'$ end
              update cooling schedule parameters $L, t_0, t_f, \beta$;
              $t := t_0$; /* reset the current temperature */
              apply $CRAFT$ to $\pi^*$
          end
          else $t := t(1 + \beta t)$   /* decrease the current temperature */
    end /* main loop */
end
```

Figure 8. Pseudo-code of the simulated annealing algorithm for the QAP. Here $\Delta z(\pi, i, j)$ is calculated according to (1.3).

5.3. Hybridization scheme

The hybridization scheme used is as follows. At the beginning, an initial solution is generated in a random way with the subsequent improving by means of the simulated annealing algorithm (SA-QAP). Then, an iterative hybrid process starts. It consists

```
procedure TS-QAP /* tabu search algorithm for the QAP */
    /*input: $\pi$ – the current permutation, $n$ – the problem size */
    /*      $Q_{TS}$ – the number of iterations ($Q_{TS} \geq 1$), */
    /*      $h_{min}, h_{max}$ – lower and higher tabu list sizes ($h_{min} < h_{max}$) */
    /* output: $\pi^*$ – the best permutation found */
    $\pi^* := \pi$;
    for $i := 1$ to $n-1$ do for $j := i+1$ to $n$ do
      calculate $\delta_{ij} = \Delta z(\pi, i, j)$;
    $T := 0,\ i := 1,\ j := 1$;
    for $q := 1$ to $Q_{TS}$ do begin /* main loop */
      if $q\ mod\ 2h_{max} = 1$ then $h := randint(h_{min}, h_{max})$;
      $\Delta_{min} := \infty$;
      for $k := 1$ to $|N_2|$ do begin /* find the best move */
        $i := iif(j < n, i, iif(i < n-1, i+1, 1)), \quad j := iif(j < n, j+1, i+1)$;
        $tabu := iif(t_{ij} \geq q, 'TRUE', 'FALSE')$,
        $aspired := iif(z(\pi) + \delta_{ij} < z(\pi^*), 'TRUE', 'FALSE')$;
        if $((\delta_{ij} < \Delta_{min})$ and $(tabu = 'FALSE'))$ or $(aspired = 'TRUE')$
        then begin
          $u := i, \quad v := j$;
          if $aspired = 'TRUE'$ then $\Delta_{min} := -\infty$ else $\Delta_{min} := \delta_{ij}$
        end /* if */
    end /* for */
    if $\Delta_{min} < \infty$ then begin
      /* perform the move: replace the current permutation by the new one */
      $\pi := \pi \oplus p_{uv}$
      for $l := 1$ to $n-1$ do for $m := l+1$ to $n$ do
      update the difference $\delta_{lm}$;
      $t_{uv} := q + h$; /* update the tabu list */
      if $z(\pi) < z(\pi^*)$ then $\pi^* := \pi$ /* save the best so far permutation */
     end
    end /* main loop */
end
```

Figure 9. Pseudo-code of the tabu search algorithm for the QAP, here $\Delta z(\pi, i, j)$ is calculated according to the formula (1.3), while the difference δ_{lm} is updated according to (1.4). The function "$randint(x, y)$" returns a random integer, uniformly distributed between x and y.

of two main phases, as mentioned in Section 2: simulated annealing (SA-QAP) and tabu search (TS-QAP). In addition, a diversification mechanism is used. The role of such a mechanism play mutations that can be seen as strings of random elementary perturbations (pairwise interchanges), like p_{ij}. The mutations may also be viewed as moves in higher-order neighbourhoods N_μ, where $2 < \mu \leq n$, here, the parameter μ is referred to as a mutation level (rate). It is obvious that the large value of μ, the stronger the mutation, and vice versa. The template of the mutation procedure based on random interchanges is presented in Figure 10.

A corresponding example is shown in Figure 11.

We can add more robustness to the mutation process by letting the parameter μ vary in some interval, say $[\mu_{min}, \mu_{max}] \subset [3, n]$. In our implementation, μ varies in the following way: at the beginning, μ is equal to μ_{min}; once the maximum value

```
procedure ri-mutation
   /* random interchanges based mutation operator (ri-mutation) for the QAP */
   /* input: $\pi$ – the current permutation, $n$ – the problem size,
   $\mu$ – the mutation lavel ($\mu \in [3, n]$) */
   /* output: $\pi$ – the mutated permutation */
   for $k := 1$ to $\mu$ do begin
   choose $i, j$, randomly, uniformly, $1 \le i, j \le n, i \ne j$
   $\pi := \pi \oplus p_{ij}$ /* interchange $i$th and $j$th elements in the current permutation */
   end /* for $k$ */
end
```

Figure 10. Pseudo-code of the random interchanges based mutation.

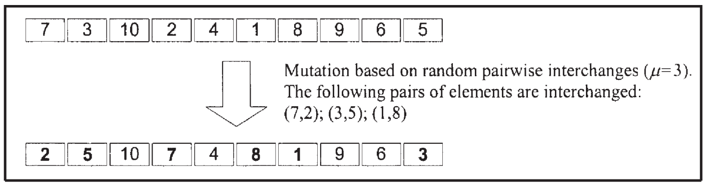
7 3 10 2 4 1 8 9 6 5
Mutation based on random pairwise interchanges (μ=3).
The following pairs of elements are interchanged:
(7,2); (3,5); (1,8)
2 5 10 7 4 8 1 9 6 3

Figure 11. Example of ri–mutation.

μ_{max} has been reached (or a better locally optimum solution has been discovered), the value of μ is immediately dropped to the minimum value μ_{min}, and so on. Note that the mutations are to be applied to the locally optimum solutions only.

The specific feature of our hybridization scheme is that if the current locally optimum solution remains unchanged for a long time (a "stagnation" of the search takes place), then a "cold restart" of the search is carried into effect. As a "cold restart", we use the construction (generation) of a new random solution coupled with the simulated annealing algorithm – the same that it used at the initialization phase. The purpose of such a restart is to add more diversity to the search, more precisely, to explore new regions of the solution space: continuing the search from the new random solution may allow to escape from a "deep" local optimum and to find better ones. The frequency of "cold restarts" is controlled by means of a special parameter, a restart interval, v, which can be related to the problem size, n, i.e. $v = \omega n$, where ω is a factor of the restart frequency ($0 < \omega < Q/n,\ Q$ is the total number of iterations of the hybridized algorithm.

The template of the resulting hybrid optimization algorithm entitled as IH-SA-TS-QAP (improved hybrid SA and TS algorithm for the QAP) is shown in Figure 12.

6. Simulation results

We have carried out a number of simulations in order to test the performance of our improved hybrid algorithm IH-SA-TS-QAP. The well-known QAP instances (test data) taken from the quadratic assignment problem library QAPLIB [9] (see also

```
procedure IH-SA-TS-QAP
  /* improved hybrid simulated annealing-tabu search algorithm for the QAP */
  /* input: $A, B$ – the connection and distance matrices, $n$ – the problem size */
  /*        $Q$ – the number of cycles (global iterations) of the hybrid algorithm */
  /*        $Q_{SA}$ – the number of iterations of the simulated annealing procedure */
  /*        $Q_{TS}$ – the number of iterations of the tabu search procedure */
  /*        $\lambda_1, \lambda_2, h_{min}, h_{max}, \mu_{min}, \mu_{max},$ – the control parameters */
  /* output: $\pi^*$ – the best permutation found */
  generate random (initial) permutation $\bar{\pi}$;
  apply SA-QAP to $\bar{\pi}$ with the parameters $Q_{SA}, \lambda_1, \lambda_2,$ and
  get the resulting permutation $\pi^{\bullet}$;
  $\pi^* := \pi^{\bullet}$;
  $q^* := 0$; /* $q^*$ is the current number of iteration at which
  the new local optimum has been found */
  $\mu := \mu_{min} - 1$; /* $\mu$ is the current mutation level */
  $v := \omega n$; /* $v$ is the restart interval (period) */
  for $q := 1$ to $Q$ do begin /* main cycle */
    apply TS-QAP to $\pi^{\bullet}$ with the parameters $Q_{TS}, h_{min}, h_{max},$ and
    get the resulting permutation $\pi^{\Delta}$;
    if $z(\pi^{\Delta}) < z(\pi^*)$ then begin
      $\pi^* := \pi^{\Delta}$, $q^* := q$, $\mu := \mu_{min} - 1$
      /* save the best so far permutation and reset the control parameters*/
    end
    if $q - q^* > v$ then begin
      generate new random permutation $\pi^{\circ}$; /* perform a "cold restart" */
      apply SA-QAP to $\pi^{\circ}$ with the parameters $Q_{SA}, \lambda_1, \lambda_2,$ and
      get the resulting permutation $\pi^{\bullet}$;
      if $z(\pi^{\bullet}) < z(\pi^*)$ then $\pi^* := \pi^{\bullet}$; /* save the best so far permutation */
      $q^* := q$, $\mu := \mu_{min} - 1$
    end
       else begin
       $\mu := iif(\mu < \mu_{max}, \mu + 1, \mu_{min})$;
       apply mutation to $\pi^*$ with the level $\mu$,
       /*i.e. perform $\mu$ random perturbations */
       and get the permutation $\tilde{\pi}$;
       $\pi^{\bullet} := \tilde{\pi}$ /* $\pi^{\bullet}$ is the mutated permutation to be processed by TS procedure */
       end
  end /* for $q$ */
end
```

Figure 12. Pseudo-code of the improved hybrid simulated annealing and tabu search algorithm for the QAP.

http://www.seas.upenn.edu/qaplib/) were used. The following algorithms were used for comparison:

1) The simulated annealing algorithm by Boelte and Thonemann (coded by the author according to the description presented in the paper of Boelte and Thonemann [6]; the algorithm is entitled as TB2M-QAP);

2) The robust tabu search algorithm by Taillard [42] (it is entitled as RTS-QAP);

3) The combined simulated annealing and tabu search algorithm by Misevičius [34] (entitled as SA-TS-QAP).

Table 1. Comparison of the algorithms for the QAP. The values of the average deviation ($\bar{\Theta}$), the percentage of 1% optimality ($P_{1\%}$), and the CPU time (in seconds) are given. The values of the best average deviations are printed in bold face.

Instance name	n	BKV	TB2M-QAP		RTS-QAP		SA-TS-QAP		IH-SA-TS-QAP		Average CPU time
			$\bar{\Theta}$	$P_{1\%}$	$\bar{\Theta}$	$P_{1\%}$	$\bar{\Theta}$	$P_{1\%}$	$\bar{\Theta}$	$P_{1\%}$	
nug30	30	6124	0.94	63	0.73	72	0.70	78	**0.52**	90	0.12
sko42	42	15812	0.66	83	1.03	55	0.55	88	**0.46**	90	0.29
sko49	49	23386	0.67	86	0.85	64	0.54	94	**0.46**	97	0.45
sko56	56	34458	0.66	84	0.95	55	0.53	93	**0.50**	96	0.65
sko64	64	48498	0.57	92	0.93	61	0.48	100	**0.45**	99	1.00
sko72	72	66256	0.60	97	0.99	53	0.52	95	**0.48**	98	1.34
sko81	81	90998	0.46	100	0.93	60	0.41	100	**0.40**	99	1.87
sko90	90	115534	0.49	99	0.96	60	**0.43**	100	**0.43**	100	2.55
sko100b	100	153890	0.39	100	0.97	61	0.34	100	**0.29**	100	3.44
sko100c	100	147862	0.46	98	1.15	36	0.34	99	**0.32**	99	3.46
sko100d	100	149576	0.49	100	0.96	57	0.43	100	**0.41**	100	3.44
sko100e	100	149150	0.52	99	1.06	45	0.45	100	**0.41**	100	3.43
sko100f	100	149036	0.54	100	0.95	62	0.46	100	**0.40**	100	3.44
tho30	30	149936	1.07	54	1.41	45	**0.90**	70	0.91	69	0.13
tho40	40	240516	1.33	31	1.30	34	1.09	46	**0.94**	54	0.26
wil50	50	48816	0.26	100	0.52	84	0.19	100	**0.16**	100	0.47
wil100	100	273038	0.25	100	0.55	69	**0.22**	100	**0.22**	100	3.45

The performance measures used are the following:

1) the average deviation from the best known solution $\bar{\Theta} = 100(\bar{z} - \tilde{z})/\tilde{z}[\%]$, where $\bar{z}$ is the average objective function value over $W = 1, 2, \ldots$ restarts (i.e. single applications of the algorithm to a given instance) and $\tilde{z}$ is the best known value (BKV) of the objective function, BKVs are from [9];

2) the percentage of solutions that are within 1% optimality $P_{1\%} = 100C_{1\%}/W$, where $C_{1\%}$ is the total number of solutions that are within 1% optimality over W restarts.

All the simulations were carried out on 300 MHz Pentium computer by using the optimization package (sub-system) *OPTIQAP* (OPTImizer for the QAP) developed by the author at Dept. of Practical Informatics of Kaunas Univ. of Technology. The computations were organized in such a way that all the algorithms use identical initial assignments and require similar CPU times (the execution time is controlled by the number of iterations). The results of the comparison, i.e. the average deviations from BKV and percentage of solutions that are within 1% optimality for each of the algorithm tested, as well as CPU times per restart are presented in Table 1. The parameters of IH-SA-TS-QAP used in simulation are as follows:

$$Q = 1,\ \ Q_{SA} = 50,\ \ Q_{TS} = 5Q_{SA} = 250\,,$$
$$\lambda_1 = 0.5,\ \ \lambda_2 = 0.05,\ \ \mu_{min} = 0.35,\ \ \mu_{max} = 0.45\,.$$

Let us note, that as long as the number of cycles, Q, is equal to 1, the parameter ω can be omitted. The number of restarts, W, is equal to 100.

Table 2. Computational results of IH-SA-TS-QAP with the various numbers of iterations. The values of the average deviation $(\bar{\Theta})$, and the CPU time (in seconds) are given. In addition, in parenthesis we give the numbers of times that BKV is found.

Instance name	$\bar{\Theta}, time$									
	1st $(W = 30)$		2nd $(W = 30)$		3rd $(W = 20)$		4th $(W = 2)$		5th $(W = 10)$	
nug30	$0.060_{[9]}$	5.0	$0.002_{[29]}$	30.0	**0**	88	**0**	150	**0**	360
sko42	$0.075_{[11]}$	10.5	$0.003_{[28]}$	63.0	**0**	180	**0**	330	**0**	720
sko49	$0.128_{[2]}$	14.9	$0.044_{[6]}$	90.0	$0.009_{[16]}$	250	**0**	470	**0**	1200
sko56	$0.168_{[1]}$	20.5	$0.049_{[3]}$	118	$0.001_{[19]}$	340	$0.001_{[14]}$	570	**0**	1380
sko64	$0.156_{[3]}$	27.4	$0.020_{[8]}$	162	$0.000_{[19]}$	460	**0**	800	**0**	1800
sko72	$0.304_{[0]}$	35.0	$0.117_{[0]}$	210	$0.012_{[0]}$	570	$0.012_{[2]}$	1020	$0.002_{[6]}$	2250
sko81	$0.191_{[0]}$	46.0	$0.074_{[0]}$	275	$0.016_{[2]}$	720	$0.013_{[2]}$	1260	$0.007_{[4]}$	2700
sko90	$0.300_{[0]}$	57.0	$0.133_{[0]}$	330	$0.042_{[1]}$	930	$0.027_{[0]}$	1440	$0.004_{[5]}$	3200
sko100a	$0.233_{[0]}$	72.0	$0.114_{[0]}$	420	$0.048_{[1]}$	1180	$0.026_{[2]}$	1980	$0.019_{[1]}$	4300
sko100b	$0.221_{[0]}$	72.0	$0.094_{[0]}$	420	$0.020_{[0]}$	1200	$0.011_{[1]}$	1950	$0.004_{[1]}$	4200
sko100c	$0.209_{[0]}$	72.0	$0.061_{[0]}$	420	$0.014_{[1]}$	1190	$0.007_{[1]}$	1940	$0.001_{[5]}$	4100
sko100d	$0.299_{[0]}$	72.0	$0.130_{[0]}$	420	$0.042_{[0]}$	1180	$0.028_{[0]}$	1960	$0.011_{[2]}$	4300
sko100e	$0.243_{[0]}$	72.0	$0.118_{[0]}$	420	$0.014_{[1]}$	1210	$0.010_{[1]}$	1980	$0.005_{[2]}$	4400
sko100f	$0.278_{[0]}$	72.0	$0.125_{[0]}$	420	$0.049_{[0]}$	1180	$0.020_{[1]}$	1970	$0.010_{[2]}$	4300
tho30	$0.074_{[22]}$	4.9	**0**	30.0	**0**	90.0	**0**	145	**0**	370
tho40	$0.196_{[1]}$	9.2	$0.023_{[6]}$	56.0	$0.012_{[6]}$	165	$0.005_{[8]}$	270	$0.002_{[8]}$	660
wil50	$0.054_{[4]}$	16.0	$0.008_{[19]}$	92	$0.002_{[19]}$	264	$0.001_{[14]}$	420	**0**	1290
wil100a	$0.175_{[0]}$	72.0	$0.084_{[0]}$	420	$0.011_{[0]}$	1200	$0.004_{[2]}$	1990	$0.001_{[4]}$	4500

It turns out that the efficiency of the algorithms depends on the QAP instance being solved. Nevertheless, the results from Table 1 show that our hybrid optimization algorithm IH-SA-TS-QAP appears to be superior to other three efficient algorithms with respect to both performance measures, especially, the average deviation. The difference in efficiency on particular instances is quite significant (see, for example, the results of RTS-QAP and IH-SA-TS-QAP obtained for the instances *sko*100a–*sko*100f, or *wil*50, *wil*100).

The results of IH-SA-TS-QAP can be improved even more by increasing the values of the control parameters Q and/or Q_{SA} (Q_{TS}) but at the cost of a longer processing time. Five long runs were carried out in order to demonstrate the improvement of the quality of solutions. At each long run, the different values of the parameters Q, Q_{SA}, Q_{TS} are used:

$$\begin{aligned}
&\text{1st run}: \; Q = 30, \; Q_{SA} = 50, \; Q_{TS} = 250, \; \omega = 0.3;\\
&\text{2nd run}: \; Q = 100, \; Q_{SA} = 50, \; Q_{TS} = 500, \; \omega = 0.3;\\
&\text{3rd run}: \; Q = 200, \; Q_{SA} = 300, \; Q_{TS} = 1000, \; \omega = 0.1;\\
&\text{4th run}: \; Q = 200, \; Q_{SA} = 500, \; Q_{TS} = 1500, \; \omega = 0.05;\\
&\text{5th run}: \; Q = 300, \; Q_{SA} = 1000, \; Q_{TS} = 3000, \; \omega = 0.03;
\end{aligned}$$

the values of the other control parameters remain the same, except the parameter ω. Table 2 shows the results obtained.

These results are very promising (see 5th column of Table 2): for small and medium instances $(n \leq 64)$ (except the instance tho40), the average deviation from the best known values of the objective function is equal to zero; while, for large instances $(n = 100)$, the deviation is less than 0.02%. It can be seen from the results of 5th run that, for all the large instances tested, at least one restart (out of ten) of IH-SA-TS-QAP resulted in finding the best known solution. Moreover, for the instances *sko*100c and *wil*100, BKV was reached 5 and 4 times, respectively. This indicates that the solutions obtained for these instances are, most likely, pseudo-optimal. To our knowledge, the pseudo-optimality of the solutions for these instances has not been reported yet in the literature. It also should be stressed that even finding BKV for these instances is quite complicated task, for example, in [16], it was reported that to improve on the best known solutions for the instances *sko*100* on SPARC 10 processor, it took almost 24 hours of computation time. In a more recent work [29], an efficient genetic algorithm could not find BKV for any of these instances. It took up to 900 seconds on DEC Alpha Server 8400 to find the solutions with the average deviation around 0.3% only.

7. Concluding remarks

The quadratic assignment problem is one of hard combinatorial optimization problems. In order to obtain near-optimal or optimal solutions for this problem within reasonable times, heuristic techniques are to be applied. One of them, an improved hybrid optimization algorithm, has been proposed in this paper.

Based on the well-known simulated annealing and tabu search approaches, as well as the intelligent hybridization strategy, we have developed an effective algorithm for the QAP – IH-SA-TS-QAP (improved hybrid SA and TS algorithm for the QAP), which is an extension of the earlier author's hybrid algorithm. IH-SA-TS-QAP is distinguished for the so-called iterative hybridization scheme – a result of the elaborations of possible hybrid heuristic paradigms.

The additional features of our algorithm are the diversification and "cold restart" mechanisms that are used in order to try to avoid a possible "stagnation" of the search. These mechanisms and the refined hybridization scheme resulted in high quality solutions obtained during the simulations with a number of the QAP instances (test data) from the QAP library – QAPLIB. These solutions indicate that, for the QAP instances examined, the proposed algorithm appears to be superior to the "pure" simulated annealing and tabu search algorithms, as well as the earlier author's hybrid (combined) SA and TS algorithm. Thus, it may be considered to be one of the most efficient single-solution algorithms for the QAP.

Regarding the future work, the emphasis on the further extensions of the proposed hybrid approach should be made. Both the elaboration of the hybridization scheme and improvements of its basic components (i.e. SA and TS procedures) are possible, for example:

a) introducing new cooling schedules for the SA algorithm;

b) applying other tabu conditions/aspiration criteria for the TS algorithm;

c) trying a more accurate adjustment (tuning) of the control parameters (the initial and final temperatures, the tabu list sizes, etc).

It might also be worthy to incorporate the proposed hybrid algorithm into other (population-based, hybrid) meta-heuristics, for example, genetic and evolutionary algorithms, as a very efficient local search procedure.

References

[1] E.H.L. Aarts and J.H.M. Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, Chichester, 1989.

[2] E.H.L. Aarts, J.H.M. Korst and P.J.M.van Laarhoven. Simulated annealing. In: E. Aarts and J.K. Lenstra(Eds.), *Local Search in Combinatorial Optimization*, Wiley, Chichester, 91 – 120, 1997.

[3] S. Amin. Simulated jumping. *Annals of Operations Research*, **86**, 23–38, 1999.

[4] G.C. Armour and E.S. Buffa. A heuristic algorithm and simulation approach to relative location of facilities. *Management Science*, **9**, 294 – 304, 1963.

[5] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, **6**, 126 – 140, 1994.

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

[7] R.E. Burkard. Quadratic assignment problems. *European Journal of Operational Research*, **15**, 283 – 289, 1984.

[8] R.E. Burkard, E. Çela, P.M. Pardalos and L. Pitsoulis. The quadratic assignment problem. In: *Handbook of Combinatorial Optimization*, volume 3, Kluwer, Dordrecht, 241 – 337, 1998.

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

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

[11] V. Cerný. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. In: *Tech. Report*, Comenius University, Bratislava, CSSR, 1982.

[12] H. Chen and N.S. Flann. Parallel simulated annealing and genetic algorithms: a space of hybrid methods. In: *Proceedings of Third Conference on Parallel Problem Solving from Nature*, Berlin, Springer, Jerusalem, Israel, 428 – 436, 1994.

[13] D.T. Connolly. An improved annealing scheme for the QAP. *European Journal of Operational Research*, **46**, 93 – 100, 1990.

[14] Z. Drezner. Heuristic algorithms for the solution of the quadratic assignment problem. *Journal of Applied Mathematics and Decision Sciences*, **6**, 163 – 173, 2002.

[15] Z. Drezner. A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 2003. (in press)

[16] C. Fleurent and J.A. Ferland. Genetic hybrids for the quadratic assignment problem. In: *Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 16. AMS, Providence, 173–188, 1994.

[17] C. Fleurent and J.A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, **63**, 437 – 461, 1996.

[18] B.L. Fox. Integrating and accelerating tabu search, simulated annealing and genetic algorithms. *Annals of Operations Research*, **41**, 47 – 67, 1993.

[19] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In: *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*, Nagoya, Japan, 616 – 621, 1996.

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

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

[22] F. Glover and M. Laguna. *Tabu search*. Kluwer, Dordrecht, 1997.

[23] M. Hanan and J.M. Kurtzberg. Placement techniques. In: *Design Automation of Digital Systems: Theory and Techniques*, volume 1. Prentice-Hall, Englewood Cliffs, 213 – 282, 1972.

[24] P. Hansen and B. Jaumard. *Algorithms for the maximum satisfiability problem*. RUTCOR Search Report 43-87, Rutgers University, USA, 1987.

[25] A. Hertz, E. Taillard and D. de Werra. Tabu search. In: *Local Search in Combinatorial Optimization*, Wiley, Chichester, 121 – 136, 1997.

[26] T.C. Hu and E.S. Kuh (Eds.). *VLSI Circuit Layout: Theory and Design*. IEEE Press, New York, 1985.

[27] S. Kirkpatrick, Jr. C.D. Gelatt and M.P. Vecchi. *Optimization by simulated annealing*, volume 220. Science, 1983.

[28] P.J.M.van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht, 1987.

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

[30] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, **34**, 111 – 124, 1986.

[31] O. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, **63**, 57 – 75, 1996.

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

[33] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A.Teller and E.Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, **21**, 1087 – 1092, 1953.

[34] A. Misevičius. Combining simulated annealing and tabu search for the quadratic assignment problem. *Information Technology and Control*, **3**(20), 37 – 50, 2001.

[35] A. Misevičius. A new simulated annealing algorithm for the quadratic assignment problem. In: *Materials of the International Conference on Production Research (ICPR-16) (Prague, Czech Republic)*, Prague, Czech Association of Scientific and Technical Societies, 2001.

[36] I.H. Osman and N. Christofides. Capacitated clustering problem by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, **1**, 317 – 336, 1994.

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

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

[39] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, **2**, 33 – 45, 1990.

[40] L. Steinberg. The backboard wiring problem: a placement algorithm. *SIAM Review*, **3**, 37 – 50, 1961.

[41] T. Stuetzle. Iterated local search for the quadratic assignment problem. Technical report, Darmstadt University of Technology, 1999.

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

[43] E.G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, **8**, 541 – 564, 2002.

[44] M. Wilhelm and T. Ward. Solving quadratic assignment problems by simulated annealing. *IIE Transactions*, **19**, 107 – 119, 1987.

[45] Q. Zeng and K.C. Mouskos. *Heuristic search strategies to solve transportation network design problem*. Tech. Report, New Jersey Dept. of Transportation and the National Center for Transportation and Industrial Productivity, USA, 1997.

Patobulintas hibridinis optimizavimo algoritmas kvadratinio paskirstymo uždaviniui

A. Misevičius

Šiame straipsnyje pasi¯ulytas patobulintas hibridinis euristinis optimizavimo algoritmas gerai žinomam, sudėtingam kombinatorinio optimizavimo uždaviniui, b¯utent, kvadratinio paskirstymo (KP) uždaviniui. Tai – pagerinta autoriaus ankstesnio hibridinio algoritmo versija. Naujasis algoritmas pasižymi tuo, jog čia išplėtota efektyvių euristikų (atkaitinimo modeliavimo (AM) (angl. simulated annealing) ir tabu paieškos (TP) (angl. tabu search) "hibridizacijos" idėja. "Hibridizacija" remiasi vadinamąja iteracine schema: TP algoritmas panaudojamas kaip postanalizės proced¯ura AM algoritmo gautajam sprendiniui, savo ruožtu, AM algoritmas taikomas sprendinių sekai, gautai sprendinių diversifikavimo/generavimo keliu. Svarbi pasi¯ulyto algoritmo savybė yra ir ta, kad jame realizuotas vadinamasis "šaltojo pakartotinio starto" principas, kurio paskirtis padėti išvengti galimų paieškos "stagnacijos" situacijų. Naujasis algoritmas išbandytas su KP uždavinio duomenimis iš testinių pavyzdžių bibliotekos QAPLIB. Gauti eksperimentų rezultatai liudija, jog nagrinėtiems KP uždavinio pavyzdžiams si¯ulomas algoritmas yra pranašesnis už ankstesnius atkaitinimo modeliavimo ir tabu paieškos algoritmus, taip pat už ankstesnį autoriaus hibridinį algoritmą.