INFORMATICA, 2003, Vol. 14, No. 4, 497–514

A Modified Simulated Annealing Algorithm for the Quadratic Assignment Problem

Alfonsas MISEVIČIUS
Department of Practical Informatics, Kaunas University of Technology
Studentų 50–400a, LT-3031 Kaunas, Lithuania
e-mail: misevi@soften.ktu.lt

Received: February 2003

Abstract. The quadratic assignment problem (QAP) is one of the well-known combinatorial optimization problems and is known for its various applications. In this paper, we propose a modified simulated annealing algorithm for the QAP – M-SA-QAP. The novelty of the proposed algorithm is an advanced formula of calculation of the initial and final temperatures, as well as an original cooling schedule with oscillation, i.e., periodical decreasing and increasing of the temperature. In addition, in order to improve the results obtained, the simulated annealing algorithm is combined with a tabu search approach based algorithm. We tested our algorithm on a number of instances from the library of the QAP instances – QAPLIB. The results obtained from the experiments show that the proposed algorithm appears to be superior to earlier versions of the simulated annealing for the QAP. The power of M-SA-QAP is also corroborated by the fact that the new best known solution was found for the one of the largest QAP instances – THO150.

Key words: heuristics, local search, simulated annealing, quadratic assignment problem.

1. Introduction

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

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

One of the applications of the quadratic assignment problem is computer-aided design, namely, the placement of electronic components into locations (positions) on a board (chip) (Steinberg, 1961; Hanan and Kurtzberg, 1972). In this context, the matrix $A = (a_{ij})_{n\times n}$ can be interpreted as a matrix of connections between components; in this case, a_{ij} is the number of the nets connecting component i and component j. The matrix $B = (b_{kl})_{n\times n}$ is a distance matrix, where b_{kl} represents the distance from location k to location l. Each placement configuration corresponds to a certain permutation, $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, where $\pi(i)$ denotes the location that component i is placed

into. Thus, z can be treated as a total estimated wire length (sum of the half-perimeters of the nets) obtained when n components are placed into n locations. The goal is to minimize the total estimated wire length. See, for example, (Burkard, 1984; Çela, 1998) for an extended list of other applications of the QAP.

The quadratic assignment problem is one of the most complex combinatorial optimization problems. It has been proved that the QAP is NP-hard (Sahni and Gonzalez, 1976). Problems of size, say $n > 30$, are not, to this date, practically solvable in terms of obtaining exact solutions (Hahn *et al.*, 1999). Therefore, heuristic approaches have to be used for solving medium- and large-scale QAPs. One such approach that has yielded promising results is simulated annealing (Burkard and Rendl, 1984; Wilhelm and Ward, 1987; Connolly, 1990; Thonemann and Bölte, 1994; Bölte and Thonemann, 1996). Other approaches for the QAP, similar to the simulated annealing, are worth mentioning: threshold accepting (Nissen and Paul, 1995), simulated jumping (Amin, 1999), intensive search (Misevičius, 2000). An exhaustive list of other heuristic methods for the QAP one can find in (Çela, 1998).

This paper is organized as follows. Sections 2, 3 survey the simulated annealing approach and its applications to the QAP. Section 4 describes a modified simulated annealing algorithm for the quadratic assignment problem. The results of the computational experiments are presented in Section 5. Section 6 completes the paper with conclusions.

2. Simulated Annealing for Combinatorial Optimization Problems

2.1. *History*

Simulated annealing originated in statistical mechanics. It is based on a Monte Carlo model that was used by Metropolis *et al*. (1953) to simulate energy levels in cooling solids (coercing solids into a low energy – highly ordered – state). Boltzmann's law was used to determine the probability of accepting a perturbation resulting in a change ΔE in the energy at the current temperature t, i.e.,

$$\mathrm{P} = \begin{cases} 1, & \Delta E < 0, \\ \mathrm{e}^{-\Delta E/C_B t}, & \Delta E \geqslant 0, \end{cases}$$

where C_B is a Boltzmann's constant. Cerný (1982) and Kirkpatrick *et al*. (1983) were the first who applied simulated annealing to solve combinatorial optimization problems. Starting from 1984, several authors applied simulated annealing to the QAP (see Section 3).

2.2. *Principle of the Simulated Annealing*

Let S be a set of solutions of combinatorial optimization problems with objective (cost) function $f\colon S \to R^1$. Furthermore, let $N\colon S \to 2^S$ 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 directly from s by an operation called a move (generally, the move follows objective function evaluation which is called a trial). The principle of the simulated annealing algorithm is simple (Kirkpatrick *et al.*, 1983): start from a random solution. Given a solution s select a neighbouring solution s' and compute the difference in the objective function values, $\Delta f = f(s') - f(s)$. If the objective function value is improved ($\Delta f < 0$), then replace the current solution by the new one, i.e., perform a move, and use resulting configuration as the starting point for the next trial. If $\Delta f \geqslant 0$, then accept a move with probability

$$\mathrm{P}(\Delta f) = \mathrm{e}^{-\Delta f/t}, \tag{2}$$

where t is the current temperature value (Boltzmann's constant is not required when applying the algorithm to combinatorial problems). The procedure is repeated until a stopping condition is satisfied, for example, a predefined number of trials has been performed (for other termination criteria, see Section 2.5). Usually, "best so far" (BSF) solution (instead of "where you are" (WYA) solution) is regarded as the result of the algorithm. Regarding the probabilistic acceptance (2), it is achieved by generating a random number in [0,1] and comparing it against the threshold $\mathrm{e}^{-\Delta f/t}$(here, the exponential function plays a role of an acceptance function).

Simulated annealing algorithms differ each from other with respect to the following factors: neighbourhood search, cooling (annealing) schedule and termination criterion.

2.3. *Neighbourhood Search*

Suppose, $S = \{s | s = (s(1), s(2), \ldots, s(n))\}$, where n is the cardinality of the set. Given a solution s from S, a k-exchange neighbourhood function $N_k(s)$ is defined as follows:

$$N_k(s) = \{s' | s' \in S,\ \ d(s, s') \leqslant k\}, \tag{3}$$

where $d(s, s')$ is the distance between solutions s and s': $d(s, s') = \sum_{i=1}^{n} \mathrm{sgn}|s(i) - s'(i)|$. If $k = 2$, one obtains 2-exchange neighbourhood function which is widely used in combinatorial problems. In this case, any neighbouring solution s' can be reached from the solution s by interchanging (displacement) exactly two elements in s.

Two alternatives exist when searching the neighbourhood. First, choose the next potential solution at random. Second, explore the neighbourhood in a systematic way having all the possible exchange elements ordered ("shuffled"). The precise order is irrelevant, it is only essential that the neighbourhood is explored thoroughly.

2.4. *Cooling Schedule*

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, and c) an equilibrium test.

The "behaviour" of the simulated annealing algorithm depends on the temperature t. Perhaps the most important thing is how the initial temperature t_0 is determinated. 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. One may choose $t_0 = \Delta f_{\max}$, where $\Delta f_{\max}$ is the maximal difference of the objective function values between any two neighbouring solutions. Exact calculation of $\Delta f_{\max}$ is quite time-consuming in many cases. Therefore, various approximations are used. The final value of the temperature t_f may be related to the smallest possible change in the objective function values between two neighbouring solutions.

The temperature is not a constant, but changes (usually, decreases) over time according to some updating function. The updating function that guarantees optimality (i.e., optimal cooling schedule) is (Anily and Federgruen, 1987; Hajek, 1988)

$$t_k = \mathrm{const}/\log(k+2), \quad k = 0, 1, \ldots. \tag{4}$$

In this case, the run time is too long for most applications. Hence, there are used heuristic schedules that cool down much more faster. The most commonly used schedules are (Kirkpatrick *et al.*, 1983)

$$t_{k+1} = \alpha \cdot t_k, \quad k = 0, 1, \ldots, \ t_0 = \mathrm{const}, \ \alpha < 1 \text{ (geometric schedule)}, \tag{5}$$

and (Lundy and Mees, 1986)

$$t_{k+1} = t_k/(1+\beta t_k), \quad k=0, 1, \ldots, \ t_0=\mathrm{const}, \ \beta \ll t_0 \text{ (Lundy–Mees schedule)}. \tag{6}$$

It should be noted that, in the state-of-the-art simulated annealing algorithms, the temperature rather changes periodically than decreases monotonically, i.e., reannealing (or tempering) – a sequence of heatings and coolings – is considered instead of the straightforward annealing. (Non-monotone cooling schedules have been introduced by Hajek and Sasaki as far back as 1989 (Hajek and Sasaki, 1989).) A variety of ways exist to reanneal (see, for example, (Osman, 1993; Bölte and Thonemann, 1996; Mann and Smith, 1996)).

Kirkpatrick *et al.* (1983) proposed that, at each temperature, the cooling schedule must allow the simulation to proceed long enough for the process to reach steady state – equilibrium. Various equilibrium tests can be used to determine if the temperature should be updated (reduced). Typically, the temperature is decreased after a fixed number of trials; as a standard, this number is proportional to the size of the neighbourhood (the cooling schedule is said to have a fixed length). More sophisticated equilibrium criteria are possible, for example, the number (ratio) of accepted and rejected pairwise interchanges (Kirkpatrick *et al.*, 1983). The number of trials at each temperature may be quite large, although the temperature steps can be relatively large also. (This is the homogeneous case of simulated annealing in analogy to homogeneous Markov chains which are used to model the simulated annealing algorithm (Seneta, 1981).) Another case is when the

temperature is reduced, but by a very small amount, after every trial; in fact, no equilibrium test is used. (This is the inhomogeneous case of annealing.) Note that formula (5) is more suitable for the homogeneous annealing, whereas formula (6) – for the inhomogeneous one. In the last case, it is easy to relate the coefficient β and the number of trials, i.e., schedule length, L, when the initial and final values of the temperature (t_0, t_f) are fixed:

$$\beta = (t_0 - t_f)/Lt_0t_f. \tag{7}$$

2.5. *Termination Criterion*

In theory the simulated annealing procedure should be continued until the final temperature t_f is zero, but in practice other stopping criteria are applied: a) the value of the objective function has not decreased for a large number of consecutive trials (it means that it has become unlikely that any improvement can be reached); 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 has been executed. In the first two cases, the simulated annealing algorithm has a non-deterministic run time, whereas the third case is the case of algorithms with a deterministic run time.

For more details about the simulated annealing, the reader is addressed to (van Laarhoven and Aarts, 1987; Aarts and Korst, 1989; Aarts *et al.*, 1997).

3. Simulated Annealing Algorithms for the Quadratic Assignment Problem

Several authors applied simulated annealing to the quadratic assignment problem, as mentioned in Introduction.

Burkard and Rendl (1984) used the homogeneous simulated annealing. In their algorithm, the process remains at temperature t_k until a fixed number of trials has been considered before "going" to the next temperature t_{k+1}. The temperature is lowered according to the formula $t_{k+1} = 10 \times (0.9)^k$(this is a variant of the geometric schedule). If all the temperatures have been used, i.e., if $k > k_{\max}$, the algorithm stops.

Wilhelm and Ward's implementation (Wilhelm and Ward, 1987) also was homogeneous, but with more sophisticated equilibrium test. One checks after a certain number of trials the fluctuations in the objective function value. If the fluctuations are small enough, equilibrium is said to be reached, and the temperature is decreased (according to the above formula). Termination criterion is as follows: stop if the number of accepted moves at three successive temperatures is less than a predetermined number.

Other researchers have tried the inhomogeneous annealing. In this case, the cooling schedule used is due to Lundy and Mees with the schedule length (the number of trials) being fixed a priori. In Connolly's algorithm (Connolly, 1990), the initial temperature is calculated according to the formula $t_0 = \Delta z_{\min} + \frac{1}{10}(\Delta z_{\max} - \Delta z_{\min})$, where $\Delta z_{\min}, \Delta z_{\max}$ are, respectively, minimal and maximal difference of the objective function values obtained after R random interchanges ($R = K/2$, where $K = |N_2| =$

$n(n-1)/2$, n is the problem size). The final temperature is set to be equal to $\Delta z_{\min}$. A simple version of the reannealing is applied in this algorithm: if a large enough number of the consecutive uphill moves are rejected, then the next uphill move is accepted, the temperature is returned to the one at which the best so far solution was found, and the search is carried out at this temperature (until a stopping condition is satisfied).

Bölte and Thonemann's algorithm (Bölte and Thonemann, 1996) is similar to the one described above, but it uses much more intelligent reannealing technique, which is referred to as oscillation. The authors have taken advantage of genetic programming to determine efficient annealing schedules for the QAP. They ascertained that good schedules have the following properties:

– the temperature at the beginning and at the end of schedule is far above zero;

– the temperature is usually not constant or monotone decreasing, but oscillated.

Bölte and Thonemann used cosine function to generate a schedule that oscillates around the temperature t with an amplitude of $0.5t$. The number of full periods the schedule contains depends on the schedule length initially chosen.

4. A Modified Simulated Annealing Algorithm for the QAP

Now we describe details of the modified simulated annealing algorithm for the QAP – M-SA-QAP.

We use random permutations as initial permutations for the simulated annealing algorithm. These permutations can be generated by a very simple procedure.

The neighbourhood function we consider is, as usual, N_2. In this case, a move from the current permutation (solution) π to the neighbouring permutation π' is formally defined by using a special operator – 2-way perturbation p_{ij}: $\Pi \to \Pi$ $(i, j = 1, 2, \ldots, n)$, which exchanges (swaps) ith and jth elements in the permutation considered (notation $\pi' = \pi \oplus p_{ij}$). Note that for a permutation π and a perturbation p_{ij}, it is more efficient to compute $\Delta z(\pi, i, j) = z(\pi \oplus p_{ij}) - z(\pi)$ than $z(\pi \oplus p_{ij})$. The direct computation of $z(\pi \oplus p_{ij})$ needs time $\mathrm{O}(n^2)$, whereas $\Delta z(\pi, i, j)$ can be calculated in $\mathrm{O}(n)$ operations:

$$\Delta z(\pi, i, j) = 2 \sum_{k=1, k\neq i,j}^{n} (a_{ik} - a_{jk}) \left(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}\right), \tag{8}$$

where $a_{ii}(b_{ii}) = \mathrm{const},\ i = 1, 2, \ldots, n$.

The neighbouring solutions are searched in a fixed order, not randomly. The order of the search is unambiguously defined by the sequence $p_{12}, p_{13}, \ldots, p_{1n}, p_{23}, \ldots, p_{2n}, \ldots, p_{n-1,n}$. $K = n(n-1)/2$ trials are needed to explore all the neighbourhood of π(in case the neighbourhood N_2 is used). We say, one iteration is performed when K trials (objective function evaluations) are made.

We use the inhomogeneous annealing schedule without any equilibrium test, as in (Connolly, 1990; Bölte and Thonemann, 1996). The temperature is decreased according to formula (6). The coefficient β is known from formula (7), because the fixed a priori schedule length (the number of trials) and the initial, as well as final temperatures

are used. In order to determine the initial and final temperatures, we use an extended approach compared to the one of Connolly. We rely on the idea that the initial and final temperature values must depend on the average (positive) difference in the objective function values (notation Δz_{avg}) – not only the minimal and maximal differences. (The differences are assumed to be evaluated by performing a fixed number of moves before starting the annealing.) Thus, our formula is as follows:

$$\begin{bmatrix} t_0 = (1 - \lambda_1 - \lambda_1')\Delta z_{\min} + \lambda_1 \Delta z_{\text{avg}} + \lambda_1' \Delta z_{\max} \\ t_f = (1 - \lambda_2 - \lambda_2')\Delta z_{\min} + \lambda_2 \Delta z_{\text{avg}} + \lambda_2' \Delta z_{\max} \end{bmatrix}, \tag{9}$$

where $\lambda_1, \lambda_1', \lambda_2, \lambda_2'$ are real numbers within the interval [0, 1] ($\lambda_1 + \lambda_1' \leqslant 1, \lambda_2 + \lambda_2' \leqslant 1$). They are referred to as temperature factors (weights). Note that, having $\lambda_1 = 0,\ \lambda_1' = 0.1,\ \lambda_2 = 0,\ \lambda_2' = 0$ we obtain Connolly's formula.

In our algorithm, we ignore the maximal difference and make the temperature to be a function of the minimal and average differences only, i.e., $\lambda_1' = \lambda_2' = 0$, hence $t_0 = (1 - \lambda_1)\Delta z_{\min} + \lambda_1 \Delta z_{\text{avg}},\ t_f = (1 - \lambda_2)\Delta z_{\min} + \lambda_2 \Delta z_{\text{avg}},\ \lambda_1 \in (0, 1],\ \lambda_2 \in [0, 1),\ \lambda_1 > \lambda_2$. In fact, the algorithm operates with these temperature factors instead of the direct temperature values. By choosing appropriate values of λ_1 and λ_2, one can control the cooling process effectively. For example, having $\lambda_2 = \text{const}$ 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.

There exist four variants in the cooling process:

- LIT–LFT ($\lambda_1 < 0.5,\ \lambda_2 < 0.5$),
- HIT–LFT–SC ($\lambda_1 \geqslant 0.5,\ \lambda_2 < 0.5,\ \lambda_1 - \lambda_2 < 0.5$),
- HIT–LFT–RC ($\lambda_1 \geqslant 0.5,\ \lambda_2 < 0.5,\ \lambda_1 - \lambda_2 \geqslant 0.5$),
- HIT–HFT ($\lambda_1 \geqslant 0.5,\ \lambda_2 \geqslant 0.5$),

where LIT – low initial temperature, LFT – low final temperature, HIT – high initial temperature, HFT – high final temperature, SC – "slow" cooling, RC – "rapid" cooling. The variant to be used can be determined by conducting several experiments (one or few small data examples are enough (see Section 5)).

Another improvement is due to modified reannealing, i.e., periodical annealing scheme which is based on a dynamic, not static, cooling schedule. Its parameters (schedule length, etc.) are adaptively changed during execution of the algorithm. We agree with Bölte and Thonemann who argue that good annealing schedules have an oscillating component (Bölte and Thonemann, 1996). Their cosine-based oscillation (COS-oscillation) is rather neighbourhood-size-dependent. We propose a Lundy–Mees-function-based oscillation (LM-oscillation) that is rather process-dependent, i.e., it depends on the former "behaviour" of the annealing.

So, let Q ($Q \geqslant 1$) be the total number of iterations. The reannealing scheme can then be described as follows: set the schedule length L to $Qn(n-1)/2$ and start with the initial temperature defined by (9), then decrease it according to (6). When $0.5|N_2| = n(n-1)/4$ consecutive moves are rejected, stop the (preliminary) cooling.

The temperature found at this moment is denoted by t^* and the corresponding trial number, i.e., actual schedule length – L^*. After cooling is stopped, the temperature is immediately increased, i.e., the system is heated up, then the annealing with new parameters starts. Additionally, a slightly modified downhill search procedure (CRAFT) (Armour and Buffa, 1963) is applied to the best solution found so far. The process is continued until a termination criterion is satisfied. All the parameters of reannealing can be collected in so-called meta-schedule table (MST) (see Tables 1a, 1b). Note that, in M-SA-QAP, the oscillation period depends on the period (actual length) of the initial schedule (schedule No. 0).

Remind that the execution of the simulated annealing algorithm is controlled by fixing a priori the number of iterations of the neighbourhood search, Q ($Q = 1, 2, \ldots$). The algorithm terminates when the current iteration (trial) number exceeds Q ($Qn(n-1)/2$).

Some authors propose to combine the simulated annealing algorithm with a post-analysis algorithm. For example, Bölte and Thonemann applied the CRAFT algorithm. We also use a post-optimization. We found the tabu search (Skorin–Kapov, 1990; Taillard, 1991) to be ideal for this purpose. Namely, the simplified version of the robust tabu search (Taillard, 1991) was used in our implementation.

The PASCAL-like notation based detailed templates of the modified simulated annealing algorithm, the CRAFT algorithm, as well as the simplified tabu search algorithm for the quadratic assignment problem are presented in Figs. 1, 2, 3.

Table 1a
Meta-schedule table for the algorithm M-SA-QAP – LM-oscillation

Schedule No.	0	i ($i \geqslant 1$)
Schedule length	$L = L_0 = Qn(n-1)/2$	$L = \min(L^*,\ L_0 - iL^*)$
Initial temperature	$t_0 = (1-\lambda_1)\Delta z_{\min} + \lambda_1 \Delta z_{\text{avg}}$	$t_0 = \text{iif}(L < n,\ t^*,\ (1+\frac{1}{3})t^*)$
Final temperature	$t_f = (1-\lambda_2)\Delta z_{\min} + \lambda_2 \Delta z_{\text{avg}}$	$t_f = \text{iif}(L < n,\ t^*,\ (1-\frac{1}{3})t^*)$
Updating function	LM ($\beta = (t_0 - t_f)/Lt_0t_f$)	CONST ($\beta = 0$)($L < n$) or LM ($L \geqslant n$)

Table 1b
Meta-schedule table for the Bölte and Thonemann's algorithm (TB2) – COS-oscillation

Schedule No.	0	1
Schedule length	$L = L_0 = Qn(n-1)/2$	$L = L_0 - L^*$
Initial temperature	$t_0 = 10^a$	$t_0 = 1.5t^*$
Final temperature	$t_f = 2$	UNDEFINED
Updating function	LM ($\beta = 0.4/L$)	COS[b]

[a] normalized data are used,
[b] see (Bölte and Thonemann, 1996) for the detailed expression.

procedure *M-SA-QAP* { modified simulated annealing algorithm for the QAP }
{ input: π – current permutation, n – problem size }
{ Q – the number of iterations ($Q \geqslant 1$)}
{ $\lambda_1,\ \lambda_2$ – initial and final temperature factors ($0 < \lambda_1 \leqslant 1,\ 0 \leqslant \lambda_2 < 1,\ \lambda_1 > \lambda_2$)}
{ output: π^* – the best permutation }
$\pi^* := \pi,\ K := n(n-1)/2$
found $\Delta z_{\min},\ \Delta z_{\text{avg}}$ by performing K random moves (starting from π)
$L_0 := QK$ {L_0 – the total number of trials (initial cooling schedule length) }
initialize cooling schedule parameters $L,\ t_0,\ t_f,\ \beta$, set t to t_0
$i := 1,\ j := 1,\ k' := \infty,\ MAXRN := K/2,\ rejected\_count := 0, oscillation :=$ **FALSE**
for $k := 1$ **to** L_0 **do begin** { main loop of the simulated annealing algorithm }
$i :=$ 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 in the objective function values }
if $\Delta < 0$ **then** *accept* := **TRUE**
else begin if *RANDOM*() $< e^{-\Delta/t}$ **then** *accept* := **TRUE else** *accept* := **FALSE end**
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 *oscillation* = **FALSE then**
if *rejected_count* $\geqslant$ *MAXRN* **then begin**
$L^* := k,\ t^* := t$
update cooling schedule parameters $L,\ t_0,\ t_f,\ \beta$, set t to t_0
oscillation := **TRUE**, $k' := 0$
apply CRAFT to π^*
end { if ... then ... }
else { nothing }
else begin { *oscillation* = TRUE }
$k' := k' + 1$
if $k' \geqslant L$ **then begin**
if $L_0 - k < L^*$ **then** update cooling schedule parameters $L,\ t_0,\ t_f,\ \beta$
set t to t_0
$k' := 0$
in case of new π^* apply CRAFT to π^*
end { if ... else ... }
if $k' > 0$ **then** $t := t/(1 + \beta t)$ { decrease the temperature }
end { for }
apply simplified tabu search to π^* with n iterations
end { *M-SA-QAP* }

Notes

1. The function "iif" is defined as follows: $\text{iif}(x, y_1, y_2) = \begin{cases} y_1, & x = \text{TRUE} \\ y_2, & x = \text{FALSE} \end{cases}$.
2. $\Delta z(\pi, i, j)$ is calculated according to formula (8).
3. The in-built function RANDOM returns an uniform random number within the interval [0,1).

Fig. 1. Template of the simulated annealing algorithm for the QAP.

```
procedure CRAFT { steepest descent procedure for the QAP in the neighbourhood N_2}
   { input: π – the current permutation, n – problem size }
   { output: π – the optimized permutation }
   not_locally_optimal := TRUE
   i := 1, j := 1, K := n(n − 1)/2{K – size of the neighbourhood N_2}
   while not_locally_optimal do begin
      delta_min := 0 { delta_min – minimum difference of the objective function values }
      for k := 1 to K do begin
         i := iif(j < n, i, iif(i < n − 1, i + 1, 1)), j := iif(j < n, j + 1, i + 1)
         delta := z(π ⊕ p_ij) − z(π)
         if delta < delta_min then begin delta_min := delta, u := i, v := j end { if }
      end { for }
      if delta_min < 0 then π := π ⊕ p_uv{ replace the current permutation by the new one }
                       else not_locally_optimal := FALSE
   end { while }
end { CRAFT }
```

Fig. 2. Template of the CRAFT (steepest descent) procedure for the QAP.

```
procedure STS-QAP { simplified tabu search algorithm for the QAP }
   { input: π – current permutation, n – problem size }
   { M – the number of iterations (M ≤ n(n − 1)/2)}
   { output: π* – the best permutation }
   π* := π, K := n(n − 1)/2
   for i := 1 to n − 1 do for j := i + 1 to n do calculate DELTA(i, j) = Δz(π, i, j)
   for i := 1 to n − 1 do for j := i + 1 to n do TABU(i, j) := FALSE { tabu list initialization }
   current_iteration := 0, i := 1, j := 1
   repeat { main loop of the tabu search algorithm }
      current_iteration := current_iteration + 1, Δ_min := ∞
      for k := 1 to K do begin {find the best move}
         i := iif(j < n, i, iif(i < n − 1, i + 1, 1)), j := iif(j < n, j + 1, i + 1)
         Δ := DELTA(i, j)
         forbidden := TABU(i, j), aspired := (z(π) + Δ < z(π*)) AND (forbidden = TRUE)
         if ((Δ < Δ_min) AND (forbidden = FALSE)) OR (aspired = TRUE) then begin
            u := i, v := j
            if aspired = TRUE then Δ_min := −∞ else Δ_min := Δ
         end { if }
      end { for }
      π := π ⊕ p_uv{ perform the move }
      if z(π) < z(π*) then π* := π { save the best permutation }
      TABU(u, v) := TRUE
      for u := 1 to n − 1 do for v := u + 1 to n do update DELTA(u, v)
   until current_iteration = M { end of main loop }
end { STS-QAP }
```

Fig. 3. Template of the simplified tabu search algorithm for the QAP.

5. Computational Experiments

We have conducted a number of computational experiments in order to test the performance of our simulated annealing algorithm – M-SA-QAP. We used the well-known problem instances taken from the quadratic assignment problem library – QAPLIB (Burkard *et al.*, 1997). All the experiments were carried out on 120MHz Pentium com-

puter by using the program OPTIQAP (OPTImizer for the QAP) developed by the author.

The other simulated annealing algorithms used in the experiments are: C-SA-QAP – Connolly's algorithm (the code **sa_qap.c** of the algorithm can be found at `http://ina.eivd.ch/Collaborateurs/etd/default.htm`), TB2 – Bölte and Thonemann's algorithm (coded by the author according to the description presented in the paper of Bölte and Thonemann). It should be noted that, in the original version of TB2, data normalization was used (t_0 and t_f are constants, namely, $t_0 = 10,\ t_f = 2$). In the author's version, normalization of the temperature is used in such a way that $t_0 = 10NF, t_f = 2NF, NF = (a_{\max} \times b_{\max})/25$, $a_{\max} = \max_{i,j} a_{ij}$, $b_{\max} = \max_{i,j} b_{ij}$ (it seems, however, that the results of both original and author's version are very similar). In all the algorithms, the execution is controlled by the number of iterations, Q.

The performance measures used are: a) the average deviation from the best known solution – $\bar{\delta}$ ($\bar{\delta} = 100(\bar{z} - \tilde{z})/\tilde{z}[\%]$, where $\bar{z}$ is the average objective function value over $W = 1, 2, \ldots$ restarts (i.e., single applications of the algorithm to a problem instance), and $\tilde{z}$ is the best known value (BKV) of the objective function); b) the percentage of solutions that are within 1% optimality – $P_{1\%}$($P_{1\%} = 100C_{1\%}/W[\%]$, where $C_{1\%}$ is the total count of solutions that are within 1% optimality over W restarts). Note, BKVs are from (Burkard *et al.*, 1997).

Firstly, we illustrate on the well-known problem instances TAI25A ($n = 25$) and NUG30 ($n = 30$) that by using the temperature calculation according to our extended formula one obtains better results (the average deviations) than by using Connolly's formula (see Table 2).

Secondly, we demonstrate on the same instances that by applying LM-oscillation the results have slightly been improved in comparison with COS-oscillation due to Bölte and Thonemann (see Table 3). The corresponding temperature curves for the instance TAI25A are depicted in Fig. 4.

Table 2

A comparison of the temperature calculation formulas ($Q = 50,\ W = 10$)
(In order to determine $\Delta z_{\min},\ \Delta z_{\text{avg}},\ \Delta z_{\max}, n(n-1)/4$ random moves are performed)

Instance name	Formula	$\bar{\delta}$									
		1	2	3	4	5	6	7	8	9	10
TAI25A	Connolly's formula	2.99	3.01	2.97	2.89	3.11	3.13	2.98	3.01	3.04	3.05
	Extended formula[a]	2.49	2.56	2.35	2.34	2.60	2.62	2.71	2.73	2.59	2.69
NUG30	Connolly's formula	1.26	1.07	1.07	1.20	1.25	1.13	1.22	1.11	1.11	1.06
	Extended formula[a]	0.90	1.00	1.04	0.88	0.95	0.92	0.97	1.01	0.90	0.91

[a] $\lambda_1 = 0.5, \lambda_2 = 0.04$.

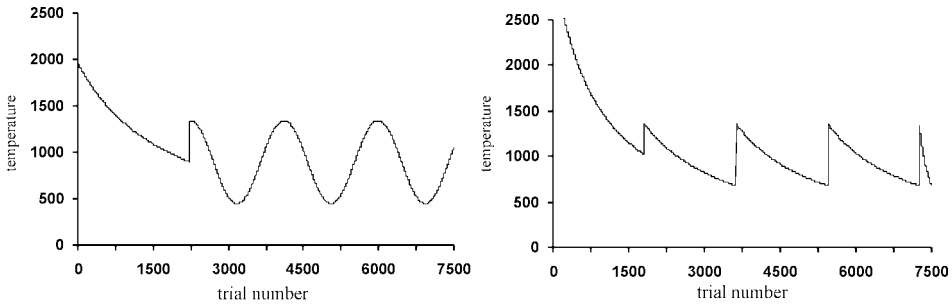
Table 3
A comparison of the oscillation rules ($Q = 50,\ W = 10$)
(Normalized initial and final temperatures are used)

Instance name	Oscillation rule	$\bar{\delta}$									
		1	2	3	4	5	6	7	8	9	10
TAI25A	COS-oscillation	2.73	2.83	2.87	2.72	2.74	2.60	2.83	2.75	2.86	2.82
	LM-oscillation	2.55	2.67	2.66	2.52	2.60	2.59	2.67	2.69	2.71	2.57
NUG30	COS-oscillation	1.06	1.08	1.05	1.02	1.04	0.94	1.17	1.02	1.05	1.02
	LM-oscillation	0.92	1.07	1.06	0.98	1.00	0.94	1.07	1.00	1.01	0.97

Then, a small experiment has been carried out in order to determine suitable values of the parameters of M-SA-QAP, namely, λ_1 and λ_2. We have chosen $\lambda_1 = 0.5$ and $\lambda_2 = 0.05$. After this, we evaluated the performance of the algorithm M-SA-QAP by comparing its solutions with those obtained by Connolly's algorithm (C-SA-QAP) and Bölte and Thonemann's algorithm (TB2), the most powerful simulated annealing algorithm for the QAP. All the algorithms use identical initial assignments and the same values of the parameters Q and W ($Q = 50$, $W = 100$). No post-analysis procedure used in C-SA-QAP; CRAFT used in TB2.

The results of the comparison, i.e., the average deviations from BKV and percentage of solutions that are within 1% optimality, as well as average CPU times per restart (in seconds), for each of the algorithm tested, are presented in Table 4 (the values of the best average deviations are printed in bold face).

It is obvious from the experiments that M-SA-QAP appears to be superior to C-SA-QAP and TB2 on the instances examined with respect to both performance measures, especially, the average deviation. The difference in performance on the particular instances (for example, STE36A–C) is really surprising. Note that the applying of the tabu search as a post-analysis procedure in M-SA-QAP increases the CPU time insignificantly due to very efficient computation of the objective function differences in the tabu search algorithm.



(**a**) the end of the curve is not at the lowest point (**b**) the end of the curve is at the lowest point

Fig. 4. The temperature curves of COS-oscillation (a) and LM-oscillation (b) for the instance TAI25A. The first curve was obtained by TB2 ($Q = 25$), the second one – by M-SA-QAP ($Q = 25$, $\lambda_1 = 0.5$, $\lambda_2 = 0.05$).

Table 4
Comparison of the algorithms

Instance name	n	BKV	C-SA-QAP			TB2			M-SA-QAP		
			$\bar{\delta}$	$P_{1\%}$	time	$\bar{\delta}$	$P_{1\%}$	time	$\bar{\delta}$	$P_{1\%}$	time
KRA30A	30	88900	4.13	2	0.65	2.65	9	0.77	**2.45**	5	0.77
KRA30B	30	91420	2.41	16	0.63	1.19	48	0.76	**1.14**	55	0.77
NUG30	30	6124	1.11	51	0.67	0.94	63	0.76	**0.85**	62	0.77
SKO42	42	15812	1.22	42	1.67	0.66	83	1.86	**0.61**	84	1.96
SKO49	49	23386	0.99	55	2.57	0.67	86	2.87	**0.57**	94	3.02
SKO56	56	34458	1.05	52	3.73	0.66	84	4.22	**0.57**	91	4.41
SKO64	64	48498	0.94	60	5.46	0.57	92	6.25	**0.50**	100	6.47
SKO72	72	66256	0.94	59	7.65	0.60	97	8.93	**0.55**	94	9.07
SKO81	81	90998	0.83	72	10.7	0.46	100	12.7	**0.43**	100	12.8
SKO90	90	115534	0.82	80	14.6	0.49	99	17.7	**0.44**	100	17.3
SKO100A	100	152002	0.79	79	19.8	0.41	100	24.6	**0.36**	100	23.6
SKO100B	100	153890	0.83	72	19.8	0.39	100	24.8	**0.35**	100	23.6
SKO100C	100	147862	0.94	58	19.8	0.46	98	24.9	**0.34**	100	23.4
SKO100D	100	149576	0.88	69	19.9	0.49	100	24.7	**0.43**	100	23.6
SKO100E	100	149150	0.97	59	19.8	0.52	99	24.4	**0.45**	100	23.6
SKO100F	100	149036	0.90	65	19.8	0.54	100	24.8	**0.47**	100	23.6
STE36A	36	9526	5.48	0	1.03	9.32	0	1.42	**2.35**	24	1.27
STE36B	36	15852	9.66	3	1.02	15.91	0	1.47	**4.06**	11	1.23
STE36C	36	8239.11	5.92	5	1.00	6.90	0	1.44	**1.89**	21	1.26
TAI25A	25	1167256	2.82	0	0.41	2.85	0	0.52	**2.32**	3	0.47
TAI30A	30	1818146	2.88	0	0.66	2.59	3	0.84	**2.23**	2	0.78
TAI35A	35	2422002	2.94	0	1.00	2.71	0	1.26	**2.37**	1	1.21
TAI40A	40	3139370	3.00	0	1.44	2.69	0	1.80	**2.45**	1	1.76
TAI50A	50	4941410	3.14	0	2.66	2.85	0	3.30	**2.60**	0	3.31
TAI60A	60	7208572	3.07	0	4.41	2.84	0	5.45	**2.56**	0	5.60
TAI80A	80	13557864	2.53	0	10.1	2.31	0	12.2	**1.96**	0	12.9
TAI100A	100	21125314	2.34	0	19.4	2.16	0	22.9	**1.82**	0	24.6
THO30	30	149936	1.85	27	0.64	1.07	54	0.74	**0.96**	70	0.76
THO40	40	240516	2.19	9	1.40	1.33	31	1.64	**1.18**	44	1.71
THO150	150	8133484[a]	1.70	4	76.0	0.56	95	110.0	**0.39**	100	88.3
WIL50	50	48816	0.46	91	2.70	0.26	100	3.03	**0.20**	100	3.19
WIL100	100	273038	0.47	100	19.8	0.25	100	24.4	**0.22**	100	23.6

[a] comes from (Amin, 1999).

The way leading to improvement of the quality of the solutions produced by M-SA-QAP is the adjustment of values of the parameters λ_1 and λ_2. The results of experiments on the instance NUG30 are presented in Table 5.

We can also improve the quality of the results by increasing the value of the parameter Q, but at the cost of a longer processing time. Seven long runs, each consisting of 100 restarts, were carried out in order to demonstrate this improvement. The long runs were

Table 5
Average deviation versus the initial and final temperature factors ($Q = 50$, $W = 100$)

λ_1	$\bar{\delta}$									
	λ_2=0	λ_2=0.01	λ_2=0.02	λ_2=0.03	λ_2=0.04	λ_2=0.05	λ_2=0.06	λ_2=0.07	λ_2=0.08	λ_2=0.09
0.1	1.19	1.15	1.18	1.03	1.02	0.93	0.84	0.86	0.81	**0.74**
0.2	1.15	1.00	0.99	0.86	0.86	0.76	0.78	0.76	0.79	**0.66**
0.3	0.90	0.77	0.83	0.84	**0.75**	**0.75**	0.77	0.76	0.79	0.80
0.4	0.97	0.80	0.79	0.86	0.79	**0.71**	0.85	0.75	0.77	0.89
0.5	0.99	0.82	0.80	0.81	0.78	0.85	**0.71**	0.84	0.75	0.80

Table 6
Computational results of M-SA-QAP with the various numbers of iterations

Instance name	$\bar{\delta}$							
	$Q = 50$	$Q = 100$	$Q = 150$	$Q = 250$	$Q = 500$	$Q = 1000$	$Q = 2000$	$Q = 5000$
KRA30A	2.45	1.84	1.58	1.19	0.83	0.53	0.33	0.12
NUG30	0.85	0.57	0.49	0.34	0.25	0.15	0.11	0.04
SKO42	0.61	0.43	0.32	0.22	0.18	0.12	0.08	0.04
STE36A	2.35	1.93	1.45	1.26	0.95	0.82	0.61	0.58
TAI25A	2.32	2.21	2.04	2.00	1.73	1.52	1.32	1.08
THO30	0.96	0.75	0.56	0.37	0.27	0.15	0.13	0.02
WIL50	0.20	0.16	0.14	0.11	0.10	0.08	0.07	0.06

Table 7
Additional results of M-SA-QAP on small QAP instances

Instance name	Time (sec)[# of restarts] needed to find BKV			
	$Q = 100$	$Q = 200$	$Q = 300$	$Q = 400$
KRA30A	$6_{[4]}$	$\mathbf{1}_{[1]}$	$8_{[2]}$	$25_{[5]}$
NUG30	$\mathbf{1}_{[1]}$	$19_{[7]}$	$8_{[2]}$	$5_{[1]}$
SKO42	$\mathbf{11}_{[3]}$	$299_{[43]}$	$18_{[2]}$	$68_{[5]}$
STE36A	$68_{[29]}$	$85_{[19]}$	$379_{[57]}$	$\mathbf{26}_{[3]}$
TAI25A	$255_{[296]}$	$170_{[103]}$	$37_{[15]}$	$\mathbf{27}_{[9]}$
THO30	$9_{[7]}$	$19_{[7]}$	$\mathbf{7}_{[2]}$	$21_{[4]}$
WIL50	$3408_{[577]}$	$787_{[69]}$	$\mathbf{320}_{[19]}$	$691_{[31]}$

organized in such a way that the control parameters are the same ($\lambda_1 = 0.5$, $\lambda_2 = 0.05$), except the number of iterations Q which increases over time. Table 6 show the results obtained.

Fig. 5 illustrates the results of fifteen long runs for the instance NUG30. Additionally, the results of four short runs are presented in Table 7.

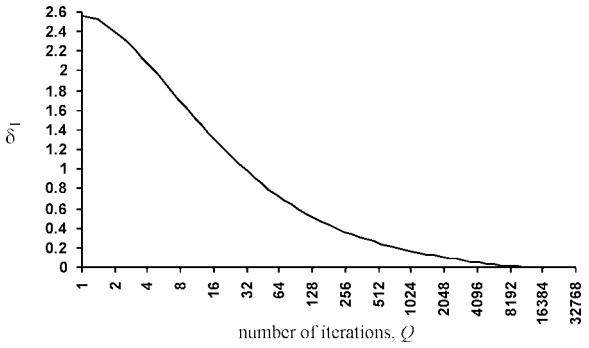


Fig. 5. Average deviation versus the number of iterations.

We would like also to stress that an improved solution has been found for the very large instance THO150 ($Q = 5000$, $\lambda_1 = 0.8$, $\lambda_2 = 0.025$). The new value of the objective function is equal to **8133398**.

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 modified simulated annealing algorithm, was proposed in this paper.

Based on the ideas of Connolly's and Bölte and Thonemann's algorithms, we developed a new modified simulated annealing algorithm for the QAP – M-SA-QAP. The novelty of the algorithm M-SA-QAP is an advanced formula of calculation of the initial and final temperatures, as well as an original cooling schedule with oscillation, i.e., periodical decreasing and increasing of the temperature. No "tuning" to each new data instance is needed, except determining once two control parameters. They are the initial and final temperature factors. Their values, we guess, are data independent and not very difficult to determine. These new features, in combination with the tabu search algorithm as a post-analysis procedure, resulted in very good solutions with moderate amount of computation time.

The results from the experiments testify that the proposed algorithm appears to be superior to the earlier algorithms of this type and should be considered to be one of the extremely efficient simulated annealing implementations for the QAP. In addition, this algorithm, like other simulated annealing algorithms, is distinguished for simple neighbourhood structure and easy programming, as well as practical realization. The only shortage is the presence of two control parameters.

An emphasis on the following directions should be made when carrying out the experiments in the future: a) using modern optimization techniques in order to find optimal values of the initial and final temperature factors, b) improving the oscillation mechanism, c) exploiting the idea of hybrid simulated annealing and tabu search approach, d)

incorporating the new simulated annealing algorithm into genetic (memetic) algorithms as a very efficient local search procedure.

Acknowledgements

Author is grateful to professor U.W. Thonemann for the comments that helped coding the algorithm TB2. He also would like to thank professor E. Taillard for providing a code of the Connolly's simulated annealing algorithm.

References

Aarts, E.H.L., and J.H.M. Korst (1989). *Simulated Annealing and Boltzmann Machines*. Wiley, Chichester.

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

Amin, S. (1999). Simulated jumping. *Annals of Operations Research*, **86**, 23–38.

Anily, S., and A. Federgruen (1987). Simulated annealing methods with general acceptance probability. *J. of Applied Probability*, **24**, 657–667.

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

Bölte, A., and U.W. Thonemann (1996). Optimizing simulated annealing schedules with genetic programming. *European J. of Operational Research*, **92**, 402–416.

Burkard, R.E. (1984). Quadratic assignment problems. *European J. of Operational Research*, **15**, 283–289.

Burkard, R.E., S. Karisch and F. Rendl (1997). QAPLIB – a quadratic assignment problem library. *J. of Global Optimization*, **10**, 391–403.

Burkard, R.E., and F. Rendl (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European J. of Operational Research*, **17**, 169–174.

Çela, E. (1998). *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer, Dordrecht.

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

Connolly, D.T. (1990). An improved annealing scheme for the QAP. *European J. of Operational Research*, **46**, 93–100.

Hahn, P.M., W.L. Hightower, T.A. Johnson, M. Guignard–Spielberg and C. Roucairol (2001). Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem. *Yugoslavian J. of Operational Research*, **11**, 41–60.

Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research*, **13**, 311–329.

Hajek, B., and G. Sasaki (1989). Simulated annealing: to cool it or not. *Systems and Control Letters*, **12**, 443–447.

Hanan, M., and J.M. Kurtzberg (1972). Placement techniques. In M.A. Breuer (Ed.), *Design Automation of Digital Systems: Theory and Techniques*, Vol.1. Prentice–Hall, Englewood Cliffs, N.J. pp. 213–282.

Kirkpatrick, S., C.D. Gelatt (Jr.) and M.P. Vecchi (1983). Optimization by simulated annealing. *Science*, **220**, 671–680.

Van Laarhoven, P.J.M., and E.H.L. Aarts (1987). *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht.

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

Mann, J.W., and G.D. Smith (1996). A comparison of heuristics for telecommunications traffic routing. In V.J. Rayward–Smith, I.H. Osman, C.R. Reeves and G.D. Smith (Eds.), *Modern Heuristic Search Methods*. Wiley, Chichester. pp. 235–254.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller (1953). Equation of state calculation by fast computing machines. *J. of Chemical Physics*, **21**, 1087–1092.

Misevičius, A. (2000). An intensive search algorithm for the quadratic assignment problem. *Informatica*, **11**, 145–162.

Nissen, V., and H. Paul (1995). A modification of threshold accepting and its application to the quadratic assignment problem. *OR Spektrum*, **17**, 205–210.

Osman, I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, **41**, 421–451.

Sahni, S., and T. Gonzalez (1976). P-complete approximation problems. *J. of ACM*, **23**, 555–565.

Seneta, E. (1981). *Non-Negative Matrices and Markov Chains*. Springer, New York.

Skorin–Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA J. on Computing*, **2**, 33–45.

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

Taillard, E. (1991). Robust taboo search for the QAP. *Parallel Computing*, **17**, 443–455.

Thonemann, U.W., and A. Bölte (1994). An improved simulated annealing algorithm for the quadratic assignment problem. Working Paper, University of Paderborn, Germany.

Wilhelm, M., and T. Ward (1987). Solving quadratic assignment problems by simulated annealing. *IIE Trans*. **19**, 107–119.

A. Misevičius was born in 1962, in Marijampole, Lithuania. He received Dipl. Eng. degree from Kaunas Polytechnic Institute, Lithuania, in 1986. From 1986 to 1991 he worked as a junior research fellow at Comput. Mach. Lab., Kaunas Polytech. Instit. 1991–1996 – doctoral student, Kaunas University of Technology. A. Misevičius got Doctor degree in 1996, Kaunas Univ. Technol. He was conferred 3rd award in Young Scientists' Competition, Kaunas Univ. Technol, in 1997.

A. Misevičius is currently an assoc. prof. at Dept. of Practical Informatics, Kaunas Univ. Technol. Author and co-author of over 30 res. papers and acad. texts on different topics of computer science. The main research interests include: computer-aided design, design and applications of heuristics and meta-heuristics for combinatorial optimization problems, theory of randomness.

Modifikuotas atkaitinimo modeliavimo algoritmas kvadratinio paskirstymo uždaviniui

Alfonsas MISEVIČIUS

Šiame straipsnyje aprašomas algoritmas, pagrįstas atkaitinimo modeliavimo (AM) (angl. *simulated annealing*) principais. Pateikiama efektyvi modifikuoto AM algoritmo versija, kuri sėkmingai pritaikyta sprendžiant vieną iš labai sunkių kombinatorinio optimizavimo uždavinių – būtent kvadratinio paskirstymo (KP) (angl. *quadratic assignment*) uždavinį.

Naujosios siūlomo algoritmo savybės yra tokios. Pirma, pasiūlyta patobulinta pradinės ir galutinės atkaitinimo temperatūros skaičiavimo formulė, kuri pasirodė efektyvesnė palyginti su iki tol plačiai naudota formule, pasiūlyta Connoly. Antra, išbandyta išplėtota atkaitinimo schema su osciliacija, t.y., periodišku temperatūros mažinimu ir didinimu (tai vadinamoji re-atkaitinimo schema (angl. *re-annealing*)). Pasiūlyta osciliacija (ji pavadinta LM-osciliacija, nes remiasi Lundy ir Mees formulės panaudojimu) yra ne statinė, o dinaminė, jos parametrai keičiami algoritmo vykdymo eigoje; kitaip tariant, osciliacija yra „priklausoma nuo proceso“, t.y., nuo vykusio atkaitinimo pobūdžio. Vėlgi, ši osciliacija pasirodė pranašesnė už anksčiau pasiūlytą (Bölte ir Thonemann'o) osciliaciją. Trečia, svarbus patobulinimas susijęs su naujos strategijos – atkaitinimo modeliavimo, kombinuojamo su tabu paieška, – išbandymu. Pasirodo, kad tabu paieškos panaudojimas „post-analizės“ procedūros vaidmenyje leidžia žymiai pagerinti atkaitinimo modeliavimo fazėje gautų sprendinių kokybę, tik nežymiai padidėjant algoritmo vykdymo laikui.

Naujasis algoritmas išbandytas su įvairių tipų KP uždavinio testiniais pavyzdžiais (duomenimis) iš KP uždavinio duomenų bibliotekos – QAPLIB. Atliktų eksperimentų rezultatai liudija, jog pasiūlytas algoritmas yra pakankamai efektyvus, pranokstantis ankstesnius pripažintus AM algoritmus KP uždaviniui. Šio algoritmo efektyvumą liudija tas faktas, jog buvo surastas naujas geriausias žinomas pasaulyje sprendinys vienam iš didžiausių KP uždavinio testinių pavyzdžių – tho150.

Pasiūlytą modifikuotą AM algoritmą tikslinga išbandyti kitose meta-euristikose, pvz., hibridiniuose genetiniuose algoritmuose, kur šis algoritmas galėtų vaidinti labai efektyvios lokaliosios paieškos procedūros vaidmenį. Taip pat šį algoritmą, su nežymiomis korekcijomis, būtų galima panaudoti kitiems sunkiems kombinatorinio optimizavimo uždaviniams.