BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 65, No. 4, 2017
DOI: 10.1515/bpasts-2017-0056

A comparison of nature inspired algorithms for the quadratic assignment problem

W. CHMIEL*, P. KADŁUCZKA, J. KWIECIEŃ, and B. FILIPOWICZ

AGH University of Science and Technology, 30 Mickiewicza Ave., 30-059 Krakow, Poland

Abstract. This paper presents an application of the ant algorithm and bees algorithm in optimization of QAP problem as an example of NP-hard optimization problem. The experiments with two types of algorithms: the bees algorithm and the ant algorithm were performed for the test instances of the quadratic assignment problem from QAPLIB, designed by Burkard, Karisch and Rendl. On the basis of the experiments results, an influence of particular elements of algorithms, including neighbourhood size and neighbourhood search method, will be determined.

Key words: permutation problem, quadratic assignment problem, ant algorithm, bees algorithm.

1. Introduction

The quadratic assignment problem (QAP) is one of the most difficult combinatorial problems, known in literature as the assignment problem with quadratic cost function or quadratic objective function. This discrete problem is one of the most fundamental issues which are the subject of the operation research. QAP generalizes a large number of theoretical issues such as the graph partitioning, maximal clique, linear arrangement problem. It models several practical problems, such as balancing of jet turbines, less-than-truckload (LTL), very-large-scale integration (VLSI), backboard wiring problem, molecular fitting. QAP belongs to the class of NP-hard problems. This is the reason why the approximation algorithms are used for the instances bigger than 30 [1–3].

In recent years the nature inspired metaheuristics have been used successfully to solve many optimization problems. Although most of them do not ensure obtaining optimal solutions, they provide good results at a reasonable time. The achievement of a result in real time is often more desirable than looking for the best result in a very long time. The implementation of the algorithms and the choice of parameters, including the neighbourhood structure, determine the algorithms' effectiveness. In the paper two nature-inspired algorithms to solve QAP are presented and tested.

The paper is organized as follows: Section 2 describes the complexity and model of QAP. Additionally, a brief description of QAP applications is presented. In Section 3, a survey on methods to the QAP problems is described. Section 4 discusses a framework of two nature-inspired algorithms with their adaptation to the QAP problems. The results of performed experiments are presented in Section 5. Finally, Section 6 provides the conclusions of the paper.

2. Quadratic assignment problem

2.1. Complexity of QAP problem. The quadratic assignment problem was introduced by Koopmans and Beckman in 1957 as a mathematical model of assigning a set of economic activities to a set of locations. In 1976, Sahni and Gonzalez proved that QAP is strongly NP-hard [4, 5], though showing that the existence of a polynomial time algorithm for solving QAPs with the entries of the coefficient matrices belonging to $\{0, 1, 2\}$ implies the existence of a polynomial time algorithm for an NP-complete decision problem. The problem used in this proof was NP-complete decision problem called the Hamiltonian cycle problem (HC).

The QAP is an NP-hard problem and this difficulty is not restricted only to finding the optimal solution. Sahni and Gonzalez [4] proved that even finding an ε-approximation solution for QAP is a hard problem in this sense that the existence of a ε-approximation algorithm implies $P = NP$.

Finding an optimal solution to QAP is a difficult task not only in case of looking for the best solution among all the feasible ones. It might appear that finding an optimal solution in the subset of the feasible solutions can be easier. For QAP it was proven that finding an optimal solution in case of the local search is a difficult problem too. Johnson and Papadimitriou in [6] created the base for the complexity theory in the local search case, where a special structure of neighbourhood is introduced. They define the PLS-problems (polynomial-time local search problem) as a set for which a locally optimal solution can be found in polynomial time. Next, they introduce a PLS-complete decision problem as an analogy of NP-complete one, which are the most difficult problems in PLS.

Murthy, Pardalos and Li [7] proposed a neighbourhood structure for QAP problem and proved that the corresponding local search problem is PLS-complete. The proposed structure is similar to that proposed by Kernighan and Lin [8] for the graph partitioning problem called *K-L* type neighbourhood structure $N_{K\text{-}L}$. As the problem of finding QAP optimal solution in $N_{K\text{-}L}$ (called (QAP, $N_{K\text{-}L}$)) is PLS-complete, then in the

*e-mail: wch@agh.edu.pl

Manuscript submitted 2016-11-23, revised 2017-01-23, 2017-02-20 and 2017-03-19, initially accepted for publication 2017-03-22, published in August 2017.

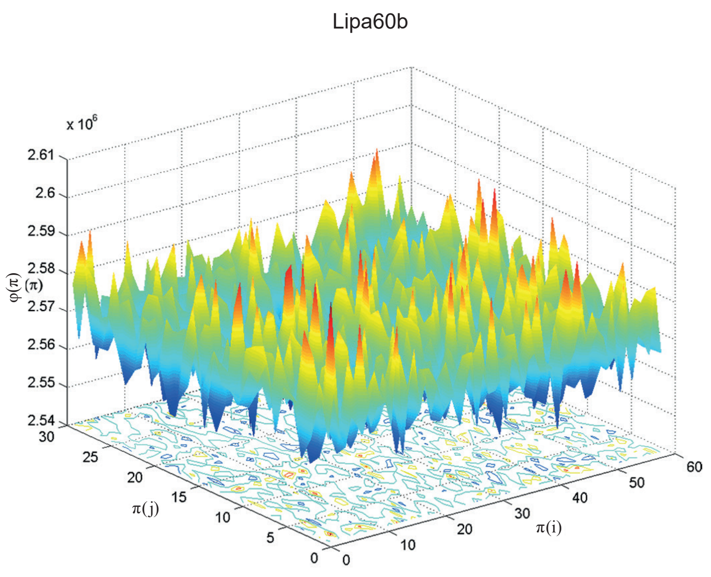

Fig. 1. An example of the landscape for QAP problem (*Lipa60*) and 2-OPT the neighbourhood structure

worst case the local search algorithm finds local minimum only after the time which is exponential in the size of the problem. The structure used in the case of other neighbourhood types, often used in QAP, is the 2-OPT (based on the pair exchange in permutation). Figure 1 shows an example of landscape for the problem instance *Lipa60b*. As could be seen, this landscape (QAP, 2-OPT) is multimodal. The neighbourhood solutions are characterised by a weak autocorrelation and hence this instance of QAP (and mostly others) is difficult to optimise.

Schaffer and Yannakakis [9] proved that the graph partitioning problem with a neighbourhood structure 2-OPT is PLS-complete. An existing PLS-reduction graph partitioning problem to QAP implies that the problem (QAP, 2-OPT) is also PLS-complete.

Several approximation algorithms for QAP use procedures based on local search, but on the basis of the above considerations, it can be proved that in the general case this approach does not guarantee finding the good solution.

There are several problems being specializations of this problem, such as GPP (graph partitioning problem), max-clique, TSP (traveling salesman problem), LAP (linear arrangement problem), backboard wiring problem, minimum weight feedback arc set and graph packing problem, and a generalization like GQAP (generalized quadratic assignment problem) that allows multiple facilities to be assigned to a single location as long as the capacity of the location allows for that. The other problems are that of Lower QAP [10] and BiQAP.

2.2. Koopmans-Beckmann quadratic assignment problem. For the given set $N = \{1, \ldots, n\}$ we define three non-negative matrices $D = [d_{ij}]_{n \times n}$, $F = [f_{ij}]_{n \times n}$, $B = [b_{ij}]_{n \times n}$. Letting permutation π is the solution of the QAP problem, then in the terminology of facilities location, $\pi(i) \in N$ $(i = 1, \ldots, n)$ defines the index of the facility and the set N is a set of the location indexes to which the facilities are assigned. For example, the permutation $\pi = (4, 5, 2, 1, 6, 3)$ defines the assignment shown in Fig. 2.

The matrix D defines distances between locations and matrix F defines the flow (weight, number of connections) between pairs of facilities. Matrix B describes the assignment cost of the facility m to the position n. It is a linear part of the assignment cost which in most cases is omitted. The solution of QAP (also denoted as QAP(F, D, B)) can be defined in permutation form $\pi = (\pi(1), \ldots, \pi(n))$ of the set of n elements (facilities). In the Koopmans-Beckmann [11] model the purpose is to find the permutation π^* which minimizes the objective function:

$$\varphi(\pi^*) = \min_{\pi \in \Pi} \varphi(\pi) \tag{1}$$

where $\varphi(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{\pi(i)\pi(j)} d_{ij} + \sum_{i=1}^{n} b_{\pi(i)i}$

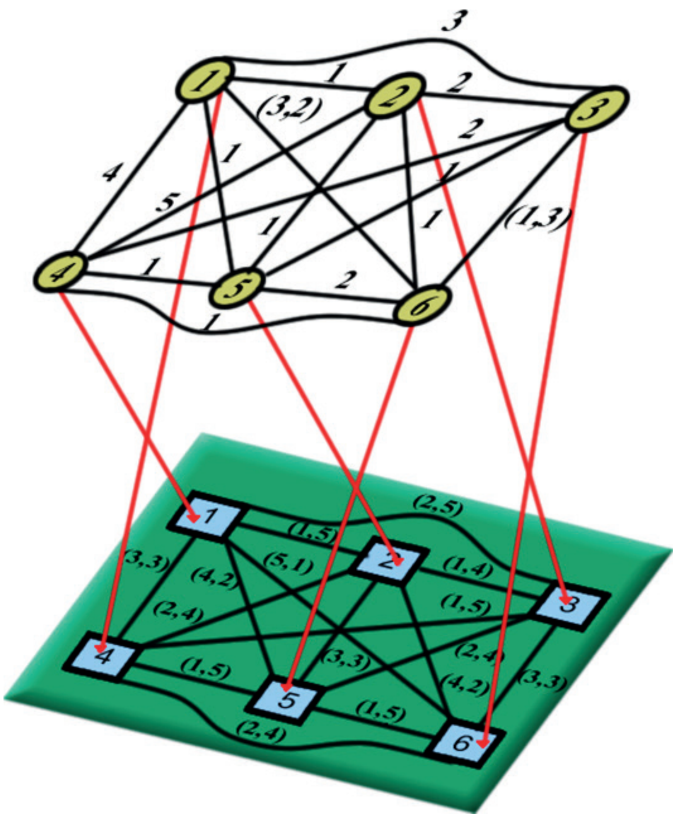

Fig. 2. Example of QAP problem: assignment of six facilities (wheels) to six locations (squares)

The objective function $\varphi(\pi)$, $\pi \in \Pi$ describes a global cost system realization and exploitation. Π is the set of permutations on the set N. In most cases matrix D is symmetric because the distance d_{ij} (between two locations i and j) is the same as d_{ji} (between j and i). Matrix F is symmetric if f_{ij} is considered as connections. If f_{ij} is flow of goods it need not be symmetric.

An example of six facilities assigned to six positions was shown in Fig. 2. In this example the flows and distances are asymmetric. For example, notation (3, 2) for facilities means that the flow from facility 1 to facility 2 is 3 and inversely 2.

The first concept on how to deal with the quadratic form was the so-called linearisation of the QAP. But, for large n the linearisation results in a large number of variables and constrains. In this case Bender's decomposition, cutting planes or other methods are not useful.

2.3. QAP applications. Nowadays the QAP problem has application in several kinds of technology in the areas like transportation [12], scheduling, electronics (wiring problem), distributed computing, statistical data analysis (reconstruction destroyed soundtrack), balancing of turbine running [13], chemistry [14], genetics [15], creating the control panels and manufacturing [16].

Figure 3 shows an example of the elastic production system (EPS) with the workstations distributed around the orbital conveyor which acts as the internal, dynamic warehouse of production system. The distances between the workstations (processing positions) spread along the conveyor depend on the transport direction.

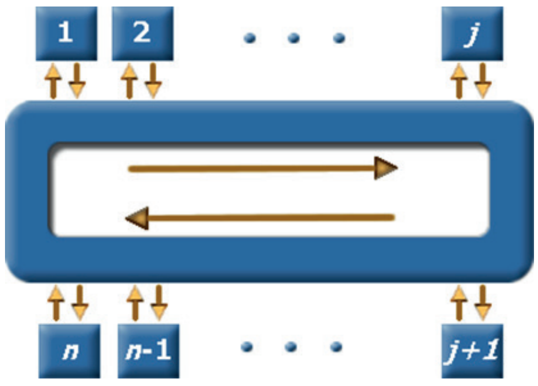

Fig. 3. Picture of the elastic production system (EPS) with the workstations distributed around the orbital conveyor

Another example of the problem defined as QAP is backboard wiring problem with a number of the modules that have to be placed on a board. The modules are pairwise connected by a number of wires. The goal is to find a placement of the modules on board so that the total length of the connected wires is minimised.

An interesting example of the problem which generalizes the QAP is the BiQAP problem. It was introduced by Burkard and Çela [17]. The definition of BiQAP was motivated by its application in the field of VLSI synthesis. This problem appears in the case of the design of synchronous sequential VLSI circuits.

3. Algorithms for QAP

Due to the quadratic assignment problem belonging to the class of NP-hard problems, many methods including exact, heuristic and metaheuristic algorithms were proposed. It should be noted that several techniques based on natural processes have been applied to the QAP. Many solution algorithms regarding the QAP can be found in literature. For a survey on these methods, one can refer to [18, 19].

Exact algorithm proposed by Roucairol [20, 21] on the basis of branch and bound method belongs to the aforementioned techniques. In this algorithm the criteria function is defined as the sum of linear components and the reduced quadratic component. This decomposition enables to determine the upper and lower bounds of the criteria function. On the basis of the lower and upper bounds, the algorithm performs an indirect search of the solution tree.

For the QAP criteria function in the form:

$$\varphi(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{\pi(i)\pi(j)} d_{ij} \tag{2}$$

the first step of the decomposition is reduction of the matrix D (distances) and the matrix F (flow), the same as in the Hungarian method for the classic (linear) assignment problem. The second step is matrix reduction which relies on rearranging the order of rows and columns due to the position of maximal element in the matrix. The connection between the value of criteria function for the QAP problem and the reduced criteria function assumes the following form:

$$\varphi(\pi) = \varphi'(\pi) + K(\pi) - \gamma \tag{3}$$

where γ is constant value obtained on the basis of reduction given as:

$$\gamma = (\textstyle\sum_i \alpha_i)(\sum_l \beta'_l) + (\sum_k \alpha'_k)(\sum_j \beta_j)$$

$\alpha_i, \beta_j, \alpha'_k, \beta'_l$ – size of reduction of i, j row/column of the matrix F and k, l row/column of the matrix D, $K(\pi) = \sum_i k_{i\pi(i)}$ – criteria function for linear assignment problem, where:

$$k_{i\pi(i)} = \alpha_i \textstyle\sum_l d_{\pi(i)l} + \beta_i \sum_l d_{l\pi(i)} + \alpha'_{\pi(i)} \sum_j f_{ij} + \beta'_{\pi(i)} \sum_j f_{ji} + \\ + \beta_i \alpha'_{\pi(i)} + \alpha_i \beta'_{\pi(i)} - (n-1)(\alpha_i \alpha'_{\pi(i)} + \beta_i \beta'_{\pi(i)}).$$

As reduced matrix F' and D' have non-negative elements and

$$\varphi(\pi) = \min_\pi K(\pi) - \gamma \tag{4}$$

then:

- lower bound is equal: $K(\pi') - \gamma$
  where $K(\pi') \geq \min_\pi \sum_i k_{i\pi(i)}$
- upper bound $\varphi(\pi') = K(\pi') - \gamma + \varphi'(\pi')$.

On the basis of the difference between the upper and the lower bounds in Roucairol branch and bound algorithm [22], a particular subset of solutions can be rejected.

Erdoğan and Tansel [23] proposed the method called branch-and-cut algorithm, based on the Koopmans – Beckmann formulation and exploit the structure of the flow and distance matrices using flow-based linearisation technique. The authors invent the two new IP formulations based on the flow-based linearisation technique that require fewer variables and yield stronger lower bounds than existing formulations. The computational experiments showed a good algorithm performance for instances with size smaller than 26.

Bashiri and Karimi in [24] compared several methods of solving the QAP problem, like the results obtained by very easy procedures including 2-OPT, 2-OPT *greedy*, 3-OPT or 3-OPT *greedy* and, even more sophisticated, the meta-heuristics as tabu search (TS), simulation annealing (SA), particle swarm optimization (PSO) and iterated fast local search (IFLS) for a large set of the instances from QAPLIB library. The average gap for heuristic and metaheuristic methods and the exact solution are slight for TS algorithm and a little bigger for SA and PSO algorithms. For small QAP instances the obtained results of PSO algorithm are worse than the results obtained by the other methods. However, the difference has disappeared for the medium and large instances. Because the authors have not described the PSO algorithm in detail, the results cannot be compared to those of the other PSO algorithms.

Congying et al. [25] used the PSO algorithm for the QAP problem. In this algorithm the great value priority method was used to transform the continuous space to discrete space. Unfortunately, the authors did not present detailed algorithm tests which were limited only to the one test instance from QAPLIB library. A modified hybrid particle swarm optimization algorithm was presented and applied to the QAP by Mamaghani and Meybodi in [26]. The hybridisation was based on joining the PSO algorithm with the hill climbing local optimization procedure. The smallest position value (SPV) rule was developed to enable the continuous version of the particle swarm optimization algorithm to be applied to the permutation problems. Another special case of the nature-inspired algorithms, called migrating birds optimization algorithm (MBO), inspired by V-formation flight of migrating birds, was proposed and tested on quadratic assignment problems by Duman et al. [27]. Chmiel et al. in [3] presented the most important properties of a multi-population genetic algorithm. These elements include: connection topology, migration size, migration interval and a method for migrant selection. A new diversity measure that is applied to permutation encoding is introduced. The proposed measure has proved effective in helping to retain balance between population diversity and convergence. A multi-population genetic algorithm, with different parameters like type of topology, migration interval, migration size and selection method was tested against several different test instances of travelling salesman problem that belongs to the NP-hard permutational problem class.

4. Bees and ant algorithms as metaheuristics based on processes found in nature

4.1. Ant algorithm. The ant algorithm was introduced by Marco Dorigo in 1992 [28] for finding the best path in the graph. It was inspired by the behavior of an ant colony from *Linepithemia humile* species. At the beginning the ants randomly check their surroundings to find the source of food, leaving a path of pheromones. Other ants follow this pheromone path. If the source of food is large, ants follow this path and reinforce the level of the pheromone trial. The probability of choosing a particular path by next ants depends on the pheromone level of the path. If the amount of food decreases, the pheromone trail leading to it weakens. The ant algorithm in most cases can be applied to graph problems. For each graph edge (i, j) a non-negative weight τ_{ij}, also called pheromone trail, is assigned. This trail is left and later read by the ants travelling between vertices. Ant k starts from vertice i and selects next vertex j from the set of neighbourhood vertices N_i. The probability of selection of the next vertices in iteration t depends on the amount of pheromones assigned to the edge (i, j) [29]:

$$p_{ij}^{k}(t) = \begin{cases} \tau'_{ij}(t) & if\ j \in N_i \\ 0 & if\ j \in N_i \end{cases} \tag{5}$$

where $\tau'_{ij}(t)$ is a normalized value of $\tau_{ij}(t)$. In each iteration an ant adds a small amount of pheromone on the edge, which is added to the current solution so that:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t-1) + \Delta\tau \tag{6}$$

The amount of the pheromone kept by the ant is constant and the strength of the pheromone trail is proportional to the quality of solution (e.g. to the length of the path). Pheromone decay runs with the introduction of a coefficient of evaporation (ρ) according to:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \Delta\tau_{ij},\ \ \rho \in (0, 1] \tag{7}$$

Once all m ants have built their solutions, the amount of the pheromone in the iteration t is given by:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \textstyle\sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{8}$$

where:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L_k & \text{if ant } k \text{ choose edge } (i, j) \\ 0 & \text{otherwise} \end{cases}$$

is the amount of pheromone left by the ant k, whereas L_k is the length of the k-ant path. QAP is the problem for which a large number of metaheuristics was created on the basis of the ant algorithm.

Max-Min Ant System (MMAS) is the metaheuristics which strongly exploits the search history by allowing only the best solution which adds the pheromone to the pheromone trail. Therefore, the modified pheromone trail is updated according to:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \Delta\tau_{ij}^{best}, \tag{9}$$

where $\Delta\tau_{ij}^{best}$ denotes the amount of pheromone left by the best solution of iteration or global best.

MMAS rather uses a simple mechanism for limiting the strengths of the pheromone trails ($\tau_{\max}$, $\tau_{\min}$ – the maximum and minimum amount of pheromone), and as a result of this, effectively avoids a premature convergence of the search process. Finally, MMAS can easily be extended by adding the local search algorithms [30].

Approximate nondeterministic tree search (ANTS) is the next algorithm in which a special formula for defining the probability distribution of each move is used.

The attractiveness of a move can be effectively estimated by means of a lower bound of the cost of the completion of a partial solution. To improve computing effectiveness, the simplified version of Gilmore and Lawler lower bound is used (called LBD). To further improve the computing effectiveness in algorithm, the simplified version of formula for defining the probability distribution at each move proposed by Colorni et al. was applied [31]. Additionally, in the algorithm a special mechanism for stagnation avoidance was proposed.

Fast Ant System (FANT) is the algorithm which incorporates a number of search strategies, such as intensification, diversification and learning mechanisms. This is realized by systematically reinforcing the impact on the search process of the best solution found so far. If the search process enters the stage of stagnation, the parameter memory is cleared to lessen the influence of the best solution [32].

It is worth noting that the selected strategy contains a set of the operations which enable an exploration and exploitation of the solution space. The proposed algorithm was created on the basis of the MMAS algorithm (see Algorithm 1). It applies the variable probability of the solutions selection used in the pheromone updating (p_{best}). At the beginning of the algorithm the best solution in the ith iteration (π_{ib}) is more preferred but the global best solution preference (π_{best}) increases with the number of the algorithm iterations.

Algorithm 1. ANT-QAP algorithm

```
Require: $\lambda$ – population size, $\varphi(\cdot)$ – criteria function, $\pi_{best}$, $\pi_{ib}$, $\rho$, $\tau_{\max}$, $\tau_{\min}$, α, $p_{Ibest}$ – initial value of the probability of choosing the $\pi_{best}$ – solution.

Step 1. Initializing population with $\lambda$ random permutation ($t = 0$):
   1. Create $\lambda$ random permutation population. Initially, each ant has assigned permutation.
   2. Evaluate fitness of the solutions in the population.
   3. Each pheromone trial $\tau_{ij}$ is set to the same non-negative value $\tau_{\max}$; $\tau_{ij}$ measures the desirability of setting $\pi_i = j$ in the solution $\pi$.
   4. Save the best solution from population: $\pi_{best} = \pi_{ib} = \min\{\pi^k\}$, $k = 1, \ldots, \lambda$.

Step 2. Create $\lambda$ solutions ($t = t + 1$):
   1. For each newly created solution, randomly choose the solution elements $\pi_i^k = j$ according to the probability $p_{ij}^k(t)$. This probability value is proportional to the normalised amount of the pheromone $\tau'_{ij}(t)$.

Step 3. Determining the best solution:
   1. For $\{\pi^k\}$, $k = 1, \ldots, \lambda$ evaluate fitness of the new solutions.
   2. Save the best solution from the new ones (iteration best) $\pi_{ib} = \min\{\pi^k\}$, $k = 1, \ldots, \lambda$ if they are better than the current best one: $\pi_{best} = \pi_{ib}$.

Step 4. Update the amount of pheromone.
   1. Update the pheromone trials. Pheromone trails are updated by taking into account the best solution produced by the search $\pi_{best}$ with probability $p_{best}$ or solutions best in the iteration – $\pi_{ib}$.
   2. Evaporate a predefined amount of pheromone
   3. Limit the pheromone trials to interval $[\tau_{\max}, \tau_{\min}]$.

Step 5. Check stop condition – a maximal number of generated solutions.
   1. If the stop condition is fulfilled, return the current best solution $\pi_{best}$ and $\varphi(\pi_{best})$.
   2. Otherwise increase probability
      $p_{best} = 1 + (p_{Ibest} - 1)e^{-\alpha t}$ and return to Step 2.
```

4.2 Bees algorithm. Bees algorithm (BA) imitates the food foraging behaviour of swarms of honey bees [33]. In its basic version, the algorithm performs a kind of the neighbourhood search combined with the random search. The colony of bees searches for the space surrounding the hive in several directions in the distance of ten kilometers. The near places with plentiful amounts of nectar or pollen are visited more frequently than other places. At the beginning of the search process, the scouts are sent from the hive into the promising paths. The scouts search randomly the space surrounding the hive and provide the information about the found food sources to the colony using waggle dance. This dance provides such information as wealth, distance and direction (relatively to the sun) to the source of food. After the dance, on the basis of food source quality and energy needed to harvest nectar or pollen, the colony of bees makes a decision about the number of bees sent to the source of food. The more bees are sent to the food source, the more effectively food will be collected. The wealth of the food source is still monitored by the returning bees. It enables them to react if the amount of food decreases. In this case the new scouts are sent to explore the space surrounding the hive to find the new promising food sources.

The bees algorithm [34, 35] can be interpreted in many ways, resulting from various implementations of the optimization algorithm based on the bees' behaviour, such as:

1. creation of the inauguration of the bees' population,
2. selection of the methods for the choice of the search direction (choice of the solution to examine),
3. definition of the number of scouts (number of examined solutions),
4. definition of the stop condition.

Implementation of the details described above and the value of the parameters determine the algorithms effectiveness. At the beginning the bees' population is created randomly (in the presented experiments all compared algorithms use the same inauguration population). The key elements which determine the algorithm effectiveness are the method of selecting the sites for the neighbourhood search and the size of a neighbourhood. The number of the examined solutions in the selected localizations is proportional to the quality of solutions. The population of the l_e best solutions (called the elite localizations) and the l_b good solutions are chosen from the whole population of solutions. The size of the searched neighbourhood for the elite localization is N_e and for the other good solutions is N_b. The remaining solutions with the worse quality (low value of criteria function) are ignored in the next search phase. Both l_e, l_b and N_e, N_b are the algorithm parameters. In the proposed implementation of the bees algorithm (called BA-QAP, Algorithm 2) for creating the neighbourhood solution (coded as the permutation) a genetic unary operators specialized in QAP problem were used:

- shift the randomly chosen facility to the random position. Other facilities are moved to the reverse direction in comparison with that in which the shift was made,
- shift the two randomly chosen facilities,
- cycle the shift facilities (rotation) around the randomly chosen position,
- create the next permutation in lexical order. In this case if more than one permutation is needed, the next permutation in the lexical order from the successor permutation will be created,
- create the predecessor permutation in lexical order. In this case if more than one permutation is needed, the predecessor permutation in lexical order of the last permutation should be created,
- shift the block of facilities over a random number of positions. Both the beginning and the end of the block is randomly chosen,
- reverse the block of facilities. Both the beginning and the end of the block are randomly chosen.

The percentage of the solutions created in the neighbourhood using the above described procedures is one of the algorithm parameters.

The next population of the solutions is created by choosing the best solution from the elite and good localizations. To keep the fixed size of the population, missing solutions are randomly created.

In BA-QAP algorithm special procedures for preventing stagnancy by getting stuck in the local minima have been implemented. The solution can exist in the population only by predefined number of iterations called life expectancy. If the value of this parameter is exceeded, the new solution is randomly generated and the old solution is replaced by the new one. The best solution which has been found so far is kept in the algorithm memory. The algorithm terminates after examining the predefined number of solutions.

Algorithm 2 uses the following variables:

λ	– swarm size,
l_e	– number of solution in elite (elite localization),
l_b	– number of good solutions (good localization),
N_e	– neighbourhood size for the elite localization,
N_b	– neighbourhood size for the good localization,
π_{best}	– the best solution found,
$\varphi(\cdot)$	– criteria function,
LT	– maximal lifetime of solution.

Algorithm 2. BA-QAP algorithm

```
Require $\lambda$, $l_e$, $l_b$, $N_e$, $N_b$, $\pi_{best}$, $\varphi(\cdot)$, $LT$

Step 1. Initialize population with $\lambda$ random solutions:
  1. Create $\lambda$ random population.
  2. Evaluate fitness of the solutions in the population.
  3. Sort population (from best to worse).
  4. Save the best solution: $\pi_{best} = \min\{\pi^k\}$, $k = 1, \ldots, \lambda$

Step 2. For each of $l_e + l_b$ best solutions:
  1. Define neighbourhood for processed solution $\pi$: $N(\pi)$.
  2. Choose the best solution from the neighbourhood
     $N(\pi)$: $\pi^* = \arg\max_{\pi \in N(\pi)} \varphi(\pi)$.

Step 3. Create a new population:
  1. For each $l_e + l_b$ localization choose the best solution (only
     one).
  2. Remove solutions which exist in population (swarm) longer
     than the predefined number of iterations $LT$ (maximal
     lifetime of solution).
  3. Create $\lambda - (l_e + l_b)$ solutions (missing solutions to fit population size).
  4. Sort population (from best to worse).

Step 4. If in the newly formed population there exists a solution
        with better value of criteria function than solution $\pi_{best}$,
        update $\pi_{best}$.

Step 5. Check the stop condition – a maximal number of generated solutions.
  1. If the stop condition is fulfilled satisfactorily, return $\pi_{best}$
     and $\varphi(\pi_{best})$.
  2. Otherwise, return to Step 2.
```

5. Experiments and results

During preliminary researches we were interested into testing several neighbourhood constructions and finding the best one. Therefore, six presented below methods were implemented for creating solutions in neighbourhood in *Step* 2 of the ANT-QAP algorithm (Algorithm 1) as well as *Step* 2 of the BA-QAP algorithm (Algorithm 2).

The following neighbourhood structures were implemented:

- *2-OPT* [7, 8],
- *ASSIGN* proposed in [36]:
  $ASSIGN = \left\{\pi \in S_n | \pi(2i-1) = 2i-1, i = 1, \ldots \tfrac{n}{2}\right\}$,

where a new solution is created by removing from the permutation the elements on the odd positions and reinserting the removed elements randomly in empty positions. S_n is a symmetric group on *n*.

- *TWIN*, which is defined only for even *n*:
  $TWIN = \left\{\pi \in S_n | \pi(2i-1) = 2i-1 \wedge \pi(2i) = 2i, i = 1, \ldots \tfrac{n}{2}\right\}$.
- *PYRAMID*:
  $PYRAMID = \big\{\pi \in S_n | i_1, i_2, \ldots, i_k, s, j_1, j_2, \ldots, j_{n-k-1};$
  $k \geq 0;\ i_1 < i_2, \ldots, < i_k;\ j_1 > j_2 > \cdots > j_{n-k-1}\big\}$.

- *PYRAMID-CV* proposed in [36], which consists of all permutations of the form $\pi \circ \alpha$, where π is a pyramidal permutation and where α is a rotation: $ROTATION = \{(k, k+1, \ldots, n, \ldots, 1, \ldots, k-1) | k = 1, .., n\}$.
- *TWISTED* defined by [37], where the permutations are created dividing a permutation into some sections. Afterwards, each section is twisted in random order.

All methods for creating solutions in the neighbourhood, specified above, were tested on the basis of the test instances from QAPLIB library.

The best results were obtained by using the 2-*OPT* neighbourhood structure. Therefore, all the results presented in this paper are based on this structure.

ANT-QAP and BA-QAP algorithms were implemented and tested on the set of instances of size $n = 21$–60 from the QAPLIB library developed by Burkard and Rendl [1]. The library contains several instances of QAP problems which model real problems (from architecture, keyboard developing, Manhattan streets, etc.) and the instances created only for testing the purpose with special properties.

It is worth mentioning that solving the problems of size bigger than 25 is still considered to be a difficult task. In Tables 1 and 2 the results of the two algorithms – the ant algorithm (AA) and the bees algorithm (BA) are shown, where φ_{ref} is the best known objective value from QAPLIB, φ_{best} is the best found objective value in 10 algorithm runs and $E = 100\% \frac{\varphi_{best} - \varphi_{ref}}{\varphi_{ref}}$ is a percentage relative gap of the solution obtained by our algorithms from the reference value. The referenced solutions for the problem with the size equal and below 25 are obtained using B&B Roucairol algorithm. For better comparison the presented results have been obtained using the same type of the computing unit.

Table 1
Results for the ant algorithm

Instance	φ_{ref}	φ_{best}	$E[\%]$
Chr12a	9 552	9 552	0
Chr20a	2 192	2 464	11.04
Els19	17 212 548	17 257 786	0.26
Esc32a	130	134	2.99
Esc64a	116	132	12.12
Esc128	64	78	17.95
Had20	6 922	6 922	0
Kra32	88 700	88 700	0
Lipa50a	62 093	62 773	1.09
Lipa90a	360 630	363 141	0.69
Nug20	2 570	2 570	0
Nug30	6 124	6 124	0
Rou20	725 522	725 522	0
Scr20	110 030	110 030	0
Sko42	15 812	15 940	0.8
Sko90	115 534	116 572	0.89
Sko100a	152 002	153 148	0.75
Ste36a	9 526	9 790	2.7
Tai30a	1 818 146	1 861 488	2.33
Tai50a	4 938 796	5 078 694	2.75
Tai64c	1 855 928	1 855 928	0
Tai100a	21 052 466	2 152 146	2.77
Tai100b	1 185 996 137	1 416 812 373	16.29
Tai150b	498 896 643	539 508 841	7.53
Tai256c	44 759 294	44 914 802	0.35
Tho40	240 516	240 516	0
Wil50	48 816	49 002	0.38
Wil100	273 038	274 466	0.52
Average			**3.01**

Table 2
Results for the bees algorithm

Instance	φ_{ref}	φ_{best}	$E[\%]$
Chr12a	9 552	9 552	0
Chr20a	2 192	2 444	10.31
Els19	17 212 548	17 212 548	0
Esc32a	130	134	2.99
Esc64a	116	116	0
Esc128	64	64	0
Had20	6 922	6 922	0
Kra32	88 700	88 700	0
Lipa50a	62 093	62 746	1.04
Lipa90a	360 630	363 060	0.67
Nug20	2 570	2 570	0
Nug30	6 124	6 124	0
Rou20	725 522	725 522	0
Scr20	110 030	110 030	0
Sko42	15 812	15 940	0.8
Sko90	115 534	116 418	0.76
Sko100a	152 002	15 274	0.83
Ste36a	9 526	9 776	2.56
Tai30a	1 818 146	1 861 488	2.33
Tai50a	4 938 796	5 063 482	2.46
Tai64c	1 855 928	1 855 928	0
Tai100a	21 052 466	21 624 756	2.65
Tai100b	1 185 996 137	1 339 342 571	11.45
Tai150b	498 896 643	510 917 010	2.35
Tai256c	44 759 294	44 895 140	0.3
Tho40	240 516	240 516	0
Wil50	48 816	48 994	0.36
Wil100	273 038	274 308	0.46
Average			**1.51**

In all tests the parameters of ant and bees algorithms are as follows:

- AA: $\lambda = 20$, $\rho = 0.1$, $\tau_{\min} = 0.1$, $\tau_{\max} = 25$, $\alpha = 0.05$, $p_{Ibest} = 0.1$,
- BA: $\lambda = 100$, $l_e = 35$, $l_b = 50$, $N_e = 100$, $N_b = 50$, $LT = 4$.

The program was implemented using C# language and tests were conducted using a workstation with i7–4860HQ/3.60 GHz processor and Windows 7 operation system. This enables to compare the computation efficiency (e.g. computation time) of the nature-inspired algorithm with the exact method. Other referenced solutions are obtained from QAPLIB library.

When going through the presented instances, we can see that there are two types of such solutions – the optimal and the lower bounds. In Tables 1 and 2 reference solutions are optimal or best known suboptimal solutions. In order to describe the solution of the considered cases more clearly, let us briefly review the methods of the obtained reference solutions.

During the experiments the following instances were used: *Chr12a*, *Chr20a* with the optimal solution found by the parallel branch and bound algorithm [38], *Els19* – the optimal solution was first found by Mautor using the parallel branch and bound algorithm [39], *Esc32a, Esc64a* – optimal solution was found by Nyberg, and Westerlund using a discrete reformulation that results in the MILP problem [42], *Esc128* – the optimal solution was found by Fischetti et al. [43] using MILP branch-and-cut solver from IBM ILOG Cplex 12.2 package. This instance is the largest QAPLIB instance ever solved to proven optimality, Had20 – the optimal solution was found by Brüngger et al. solved using the branch and bound algorithm based on the Hungarian method [44], *Kra32* – the optimal solution was found using the B&B algorithm by Anstreicher et al. [45], *Lipa50a*, *Lipa90a* – asymmetric instances produced by the generator with the known optimal solutions [46]. The optimal solution was found using the hybrid genetic algorithm by Ji et al.[47], *Nug20*, *Nug30* – Anstreicher and Brixius [45] found the optimal solution to $n = 30$ using a parallel B&B algorithm running on one thousand computers within a week, *Rou20* – instance produced by the generator with the known optimal solutions [22]. The optimal solution was found using B&B algorithm in [22], *Scr20* – the optimal solution of this problem was found by Mautor using the B&B algorithm [39], *Sko42*, *Sko90*, *Sko100a* – the best solution was obtained using the robust tabu search algorithm and the genetic algorithm (Sko100a) [40], *Ste36a* – the best solution was found using the tabu search algorithm by Skorin and Karpov [40], *Tai30a*, *Tai50a*, *Tai100a* – the best solution to the first instance was found by robust tabu search by Taillard [41] and for the second and third one by iterated tabu search by Misevicius [48], *Tai100b*, *Tai150b* – the best solution for both instances was found by Taillard, the first one using the robust tabu search [41] and the second one by GA [49], *Tai64c*, *Tai256c* – the best solution for the first instance was found using the B&B algorithm by Fischetti [43] and the second one using ant-system by Stützle [50], *Tho40* – the best solution was found by the simulation annealing algorithm by Bölte [51], Wil50, *Wil100* – the best solution was found by the simulation annealing algorithm by Bölte [51] and the genetic hybrids algorithm by Fleurent et al. [52].

A typical course optimization process for the ant algorithm and the bees algorithm was shown in Fig. 4 and Fig. 5. It is interesting that both algorithms during optimization process constantly improve the value of criteria function, even in the late stage of the optimization process, avoiding being caught in the local minimum. In all the cases swarm algorithms (ant and bees) found a solution near the optimal (average error 3.1% and 1.5%) in the time below 90 seconds (see Fig. 6).

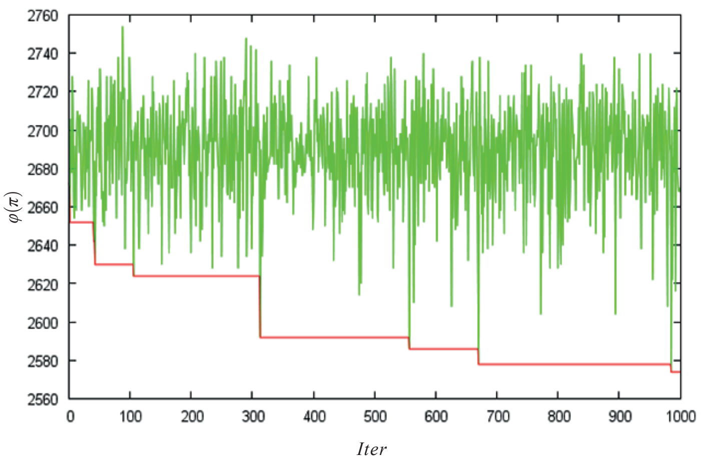

Fig. 4. Typical course of the ant algorithm for the *Nug20* instance of the QAP problem; it shows dependency between value of criteria function for the actual solution (green), the best solution (red) and the iteration number

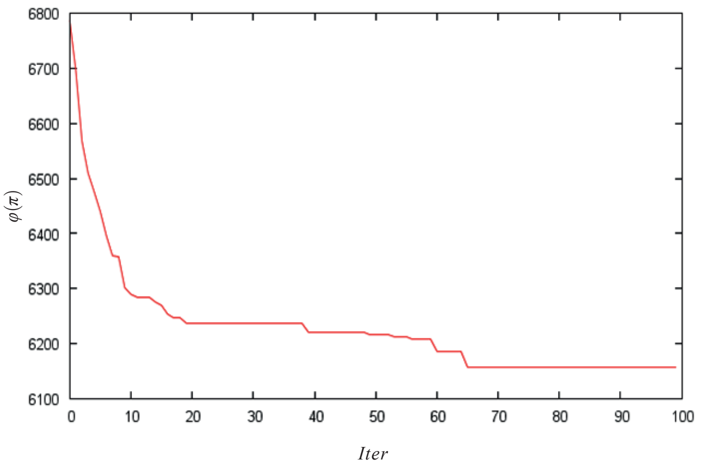

Fig. 5. Typical course of the bees algorithm for the *Nug30* instance of the QAP problem; it shows dependency between the best solution and the iteration number

For pairwise statistical comparisons of applied nature inspired metaheuristics, the Wilcoxon rank test was performed. It is a nonparametric test that aims to detect significant differences between the performances of two algorithms. Details of Wilcoxon's test can be found in [53]. All statistical experiments

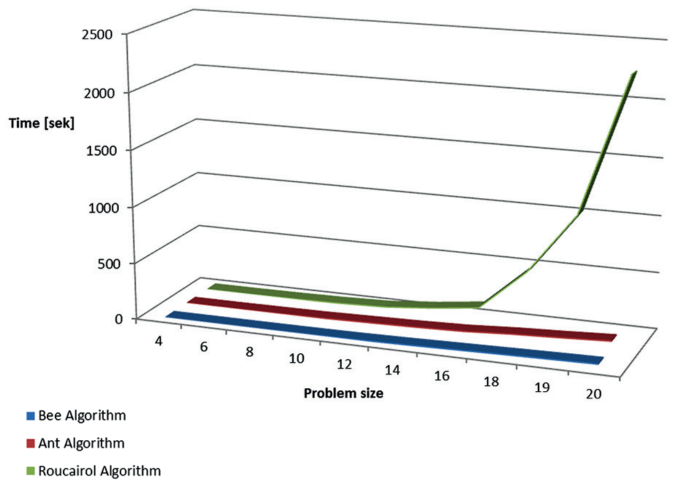

Fig. 6. Dependency between time and problem size for bees algorithm, the ant algorithm and B&B Roucairol algorithm

were conducted using the Statistica package. For the analysed instances, the value of T statistics is equal to 14, which leads to the conclusion that there are significant differences in the results obtained using the bees and the ant algorithms (with significance level below 0.05).

6. Conclusions

The results presented above enable us to state that the tested algorithms can be used in several scenarios, where the strong emphasis is put on time of obtaining good solutions which do not need to be optimal (e.g. in the on-line optimization). On the other hand, if we take under consideration the fact that the QAP problem is NP-hard, which generalizes many other discrete problems, we can anticipate that the tested algorithms can be used successfully to optimize the other NP-hard discrete problems – especially those generalized by QAP.

Acknowledgements. The authors would like to thank the reviewers for their constructive comments and suggestions.

References

[1] R. Burkard, S. Karisch and F. Rendl, "QAPLib: a quadratic assignment problem library", 1997.

[2] W. Chmiel and P. Szwed, "Bees algorithm for the quadratic assignment problem on CUDA platform", *4th International Conference on Man-Machine Interactions*, 615–625, Springer Int. Publishing (2015).

[3] W. Chmiel, P. Kadłuczka, and G. Packanik, "Performance of swarm algorithms for permutation problems", *Automatyka* 15(2), 117–126 (2009) (in Polish).

[4] S. Sahni and T. Gonzalez, "P-complete approximation problems", *J.ACM*, 23(3), 555–565 (1976).

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, 1979.

[6] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis, "How easy is local search?", *J. Comput. Syst. Sci.* 37(1), 79–100 (1988).

[7] K.A. Murthy, Y. Li, and P.M.Pardalos, "A local search algorithm for the quadratic assignment problem", *Informatica*, Vilnius, 3(4), 524–538 (1992).

[8] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", *The Bell System Technical Journal* 49(1), 291–307 (1970).

[9] A.A. Schaffer and M. Yannakakis, "Simple local search problems that are hard to solve", *SIAM J. Comput.*, 20(1), 56–87 (1991).

[10] E.L. Lawler, "The quadratic assignment problem: A brief review", *Combinatorial Programming: Methods and Applications*, 19, 351–360, Springer Netherlands 1975.

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

[12] R. Bermudez and M.H. Cole, "A genetic algorithm approach to door assignments in breakbulk terminals", *Technical Report* MBTC-1102, Mack-Blackwell Transportation Center, University of Arkansas, 2001.

[13] A. Mason and M.Rönnqvist, "Solution methods for the balancing of jet turbines", *Computers & OR* 24(2), 153–167 (1997).

[14] I. Ugi, J. Bauer, J. Brandt, J. Friedrich, J. Gasteiger, C. Jochum, and W.Schubert, "Neue Anwendungsgebiete für Computer in der Chemie", *Angewandte Chemie* 91(2), (1979) [in German].

[15] A.T. Phillips and J.B. Rosen, "A quadratic assignment formulation of the molecular conformation problem", *Journal of Global Optimization* 4, 229–241 (1994).

[16] M. Grötschel, "Discrete mathematics in manufacturing", in R.E.O. Malley (Ed.), ICIAM 1991: *Proceedings of the Second International Conference on Industrial and Applied Mathematics*, SIAM, 119–145 (1991).

[17] R.E. Burkard and E. Çela, "Heuristics for biquadratic assignment problems and their computational comparison", *European Journal of Operational Research* 83(2), 283 – 300 (1995).

[18] Z. Drezner, "The quadratic assignment problem", *Location Science* (eds.: Laport G.et al.), 345–363, Springer International Publishing (2015).

[19] E.M. Loiola, N.M.M. de Abreu, P.O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem", *European Journal of Operational Research* 176 (2), 657–690 (2007).

[20] C. Roucairol, "A reduction method for quadratic assignment problem", *Operations Research Verfahren, Methods of Operations Research* 32, 185–187 (1979).

[21] C. Roucairol, "An efficient branching scheme in branch and bound procedures", *Tims* XXVI, Masi, Universite Paris 6, 42, 1984.

[22] C. Roucairol, "A parallel branch and bound algorithm for the quadratic assignment problem", *Discrete Applied Mathematics* 18(2), 211 – 225 (1987).

[23] G. Erdoğan and B. Tansel, "A branch-and-cut algorithm for quadratic assignment problems based on linearizations", *Computers&Operations Research* 34(4), 1085- 1106 (2007).

[24] M. Bashiri and H. Karimi, "Effective heuristics and meta-heuristics for the quadratic assignment problem with tuned parameters and analytical comparisons", *Journal of Industrial Engineering International* 8(6), 1–9 (2012).

[25] L. Congying, Z. Huanping, and Y. Xinfeng, "Particle swarm optimization algorithm for quadratic assignment problem", *International Conference on Computer Science and Network Technology*, 3, 1728–1731 (2011).

[26] A. Mamaghani and M. Meybodi, "Solving the quadratic assignment problem with the modified hybrid PSO algorithm", *International Conference on Application of Information and Communication Technologies*, 1–6 (2012).
[27] E. Duman, M. Uysal and A.F. Alkaya, "Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem", *Information Sciences*, 217, 65–77 (2012).
[28] M. Dorigo, "Optimization, learning and natural algorithms", *PhD Thesis*, Politecnico di Milano, Italy (1992).
[29] E. Bonabeau, M. Dorigo, and G. Theraulaz, *From Natural to Artificial Swarm Intelligence*, Oxford University Press, 1999.
[30] T. Stützle and H. Hoos, "Max-min ant system", *Future Generation Computer Systems* 16(8), 889–914 (2000).
[31] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies", *European Conference on Artificial Life*, 134–142 (1991).
[32] E.D. Taillard, "Fant: Fast ant system", *Technical Report*, (1998).
[33] D.T. Pham and M. Castellani, "Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms", *Soft Computing* 18(5), 871–903 (2014).
[34] C.S. Chong, A.I. Sivakumar, M.Y.H. Low, and K.L. Gay, "A bee colony optimization algorithm to job shop scheduling", *Proceedings of the 38th Conference on Winter Simulation*, Monterey, (2006).
[35] B. Filipowicz and J. Kwiecień, "Swarm algorithms in optimization of quadratic assignment problem (QAP)", *Automatyka* 15(2), 159–166 (2011) (in Polish).
[36] V. Saravanov and N. Doroshko, "The approximate solution of the traveling salesman problem by a local algorithm that searches neighborhoods of factorial cardinality in cubic time". *Software: Algorithms and Programs* 31, 11–13 (1981).
[37] F. Aurenhammer, "On-line sorting of twisted sequences in linear time", *BIT* 28, 194–204 (1988).
[38] N. Christofides and E. Benavent, "An exact algorithm for the quadratic assignment problem on a tree", *Operations Research* 37(5), 760–768 (1989).
[39] T. Mautor, "Contribution a la resolution des problemes dimplanation: algorithmes sequentiels et paralleles pour laffectation quadratique", PhD thesis (1992).
[40] J. Skorin-Kapov, "Tabu search applied to the quadratic assignment problem", *ORSA Journal on Computing* 2(1), 33–45 (1990).
[41] E.D. Taillard, "Comparison of iterative searches for the quadratic assignment problem", *Location Science* 3(2), 87–105 (1995).
[42] A. Nyberg and T. Westerlund, "New exact discrete linear reformulation of the quadratic assignment problem", *Technical Report*, Abo Akademi University (2011).
[43] M. Fischetti, M. Monaci, and D. Salvagnin, "Three ideas for the quadratic assignment problem". *CPAIOR* 2011, ZIB report 11–20, 9/13 (2011).
[44] A. Brüngger, A. Marzetta, J. Clausen, and M. Perregaard, "Joining forces in solving large-scale quadratic assignment problems in parallel", *Proceedings of the 11th International 9Symposium on Parallel Processing*, 418–426 (1997).
[45] K. Anstreicher, N. Brixius, J.-P. Goux, and J. Linderoth, "Solving large quadratic assignment problems on computational grids", *Mathematical Programming* 91(3), 563–588 (2002).
[46] Y. Li and P.M. Pardalos, "Generating quadratic assignment test problems with known optimal permutations", *Computational Optimization and Applications* 1(2), 163–184 (1992).
[47] P. Ji, Y. Wu, and H. Liu, "A solution method for the quadratic assignment problem (QAP)", *The 6th International Symposium on Operations Research and Its Applications*, Xinjiang, China, 106–117, (2006).
[48] A. Misevicius, "An implementation of the iterated tabu search algorithm for the quadratic assignment problem", *OR Spectrum*, 34(3), 665–690 (2012).
[49] E.D. Taillard and L.M. Gambardella, "Adaptive memories for the quadratic assignment problem", 1997.
[50] T. Stützle, "MAX-MIN ant system for quadratic assignment problems", *Research Report AIDA*–97–04, Department of Computer Science, Darmstadt University of Technology, Germany (1997).
[51] A. Bölte and U.W. Thonemann, "Optimizing simulated annealing schedules with genetic programming", *European Journal of Operational Research* 92(2), 402 – 416 (1996).
[52] C. Fleurent, Jacques, and J.A. Ferland, "Genetic hybrids for the quadratic assignment problem", *DIMACS Series in Mathematics and Theoretical Computer Science*, American Mathematical Society, 173–187 (1993).
[53] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm Evol. Comput.* 1, 3–18 (2011).