Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

How to start a heuristic? Utilizing lower bounds for solving the quadratic assignment problem

Radomil Matousek[a], Ladislav Dobrovsky[a] and Jakub Kudela[a*]

[a]Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Brno, Czech Republic

C H R O N I C L E

Article history:
Received July 30 2021
Received in Revised Format September 21 2021
Accepted December 31 2021
Available online December, 31 2021

Keywords:
Heuristics
Lower bounds
Metaheuristics
Quadratic assignment problem
Starting values

A B S T R A C T

The Quadratic Assignment Problem (QAP) is one of the classical combinatorial optimization problems and is known for its diverse applications. The QAP is an NP-hard optimization problem which attracts the use of heuristic or metaheuristic algorithms that can find quality solutions in an acceptable computation time. On the other hand, there is quite a broad spectrum of mathematical programming techniques that were developed for finding the lower bounds for the QAP. This paper presents a fusion of the two approaches whereby the solutions from the computations of the lower bounds are used as the starting points for a metaheuristic, called HC12, which is implemented on a GPU CUDA platform. We perform extensive computational experiments that demonstrate that the use of these lower bounding techniques for the construction of the starting points has a significant impact on the quality of the resulting solutions.

1. Introduction

The NP-hard quadratic assignment problem (QAP) in its Koopmans and Beckmann form (Koopmans & Beckmann, 1957), which is notoriously difficult in practice, can be described as follows (Cela, 2013): The problem is structured on a complete directed graph $G = (V, A)$ with n nodes and n^2 arcs, together with a set of n facilities that have to be assigned to the nodes. The indices $i, j \in V$ correspond to the nodes, the indices $f, g \in N = \{1, \dots, n\}$ correspond to the facilities, $b_{i,j} \geq 0$ is a given (directed) distance from node i to node j, $a_{f,g} \geq 0$ is a given flow from facility f to facility g. By using binary variables $x_{i,f} = 1$ if facility f is assigned to node i, and 0 otherwise, the QAP can be stated as the following quadratic 0-1 optimization problem:

$$\min \quad \sum_{i\in V}\sum_{f\in N}\sum_{j\in V}\sum_{g\in N} a_{f,g} b_{i,j} x_{i\,f} x_{j,g} \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in V} x_{i,f} = 1 \quad \forall f \in N \tag{2}$$

$$\sum_{f\in N} x_{i,f} = 1 \quad \forall i \in V \tag{3}$$

$$x_{i\,f} \in \{0\ ,1\} \quad \forall i \in V\ \forall f \in N, \tag{4}$$

It is quite well known that the constraint matrix, defined by (2)-(3) is totally unimodular, implying that the optimization of any linear objective function over the QAP feasible set is just a relatively easy linear programming problem, known as the linear assignment problem (LAP) (Burkard et al., 2012).

* Corresponding author
E-mail: Jakub.Kudela@vutbr.cz (J. Kudela)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.12.003

Despite its rather simple definition, QAP is among the most difficult optimization problems that arise in practice – campus planning problem (Dickey & Hopkins, 1972), backboard wiring problem (Steinberg, 1961), hospital layout problem (Helber et al., 2016), airport gate assignment (Haghani & Chen, 1998), turbine runner in electricity generation (Laporte & Mercure, 1988), statistical analysis (Hubert & Schultz, 1976), and optimal placing of letters on touchscreen devices (Dell'Amico et al., 2009) have all been modeled as a QAP. There are several other well-known combinatorial optimization problems which can be formulated as QAPs with specific coefficient matrices (Cela, 2013) – e.g., the travelling salesman problem, graph partitioning and maximum clique, the linear arrangement problem, and packing problems in graphs, to name a few. An intriguing feature of the QAP is that even for some problems of size $n \leq 50$, such as sko42 or tai30a from the QAPLIB problem library (Burkard et al., 1997), the optimal solution is still not confirmed. Even finding an ε-optimal solution is a difficult problem. There are, however, several QAP instances/structures for which the optimal solution is attainable in polynomial time (Cela et al., 2018) or which were generated in such a way that the optimal solution is known (Li & Pardalos, 1992). Furthermore, several directions for enriching the QAP formulation have been proposed – among the most notable of these are the multi-objective formulation (Samanta et al., 2018; Sanhueza et al., 2017) and stochastic programming formulation (Popela et al., 2016; Matousek et al., 2017). For these reasons, the study of the QAP attracted quite a large amount of research from both mathematical programming and heuristics communities.

In this paper, we show that the approaches from these two communities can be successfully combined. We will utilize the lower-bounding techniques for the construction of advantageous starting points for a hill climbing metaheuristic and show on extensive computational experiments that starting the heuristic at these points yields significant improvements over the usual random starting points.

The remained of the article is structured as follows: In Section 2, the state-of-the-art in both mathematical programming and heuristics approaches to the QAP is reviewed. In Section 3 a suitable metaheuristic algorithm HC12 is described. Section 4 provides computational comparison of the mathematical programming approaches, and the results obtain from HC12. Conclusions and future research direction are summarized in Section 5.

2. Methods for approaching QAP

2.1. (Meta)Heuristics

Because of the computational difficulty of the QAP, myriads of heuristics were proposed to tackle this combinatorial problem. Among the first ones were simulated annealing (Burkard & Rendl, 1984), robust tabu search (Taillard, 1991) and genetic hybrids (Fleurent & Ferland, 1994) – although these are no longer the most efficient methods, they were able to find the best known solutions for some of the QAPLIB instances, that are yet to be beaten or proven optimal. The comparison between tabu search and simulated annealing based on a size of the QAP was conducted in (Hussin & Stützle, 2014). A local search heuristic called breakout local search enhanced by a Levenshtein Distance metric for checking solutions for similarity was described in (Aksan et al., 2017). The state-of-the-art in metaheuristics for the QAP includes population based memetic algorithms (Benlic & Hao, 2015), genetic algorithms (Ahmed, 2015), differential evolution (Hameed et al., 2020) and particle swarm algorithms (Hafiz & Abdennour, 2016). Hybrid algorithms, combining several heuristics and metaheuristics are also very prevalent. A hybrid teaching-learning based algorithm integrating tabu search within a swarm intelligence metaheuristic was described in (Dokeroglu, 2015). A memetic algorithm that uses a ternary tree structure for its population and the tabu search algorithm, which runs simultaneously, for its local search mechanism was proposed in (Harris et al., 2015). A parallel hybrid algorithm with three phases was proposed by (Tosun, 2015) – this algorithm initially benefits from a genetic algorithm to obtain a high-quality initial seed on which a diversification mechanism is run. Finally, this modified solution is used for a robust tabu search to find a near-optimal result. In (Abdel-Basset et al., 2018) the authors describe an algorithm integrating the whale optimization algorithm with a tabu search. A multistart hyper-heuristic algorithm on the grid is proposed in (Dokeroglu & Cosar, 2016) – it makes use of different metaheuristics (simulated annealing, robust tabu search, ant colony optimization, and breakout local search) and reports computations on a high-performance cluster with 368 cores and 736 GB of RAM.

Since it offers speed-up opportunity that can outperform current multicore processors, (Tsutsui & Fujimoto, 2009) applied Graphics Processing Unit (GPU) computation with compute unified device architecture (CUDA) to solve the QAP. In (Czapiński, 2013) is proposed a Parallel Multistart Tabu Search (PMTS) algorithm. It is implemented on a highly powerful GPU hardware intended for high-performance computing with the CUDA platform. Therefore, PMTS is shown to perform competitively with a single-core or a parallel CPU implementation on a high-end six-core CPU. Another GPU based algorithm is described in (Mohammadi et al., 2015) – a parallel genetic algorithm, that (as the authors report) can run up to 30 times faster than its serial counterpart. Finally, the bees algorithm implemented on the CUDA platform is proposed in (Chmiel & Szwed, 2016).

2.2. MIP Reformulations

One common mathematical programming approach for solving the QAP is to "linearize" it, that is, reformulate it as a pure or mixed integer linear programming problem. This was first done in (Gilmore, 1962) by replacing the terms $x_{i,f}x_{j,g}$ in the objective function by n^4 variables $y_{i,f,j,g} = x_{i,f}x_{j,g}$. This reformulation was further improved upon in (Adams & Johnson, 1994), calling it the level-1 reformulation-linearization technique (RLT-1). The reformulated problem than has the following form:

$$\min \quad \sum_{i\in V}\sum_{f\in N}\sum_{j\in V}\sum_{g\in N} a_{f,g}b_{i,j}y_{i,f,j,g} \tag{5}$$

$$\text{s.t.} \quad \sum_{i\in V} y_{i,f,j,g} = x_{j,g} \quad \forall j \in V \ \forall f,g \in N \tag{6}$$

$$\sum_{f\in N} y_{i,f,j,g} = x_{j\ g} \quad \forall i\ j \in V \ \forall g \in N \tag{7}$$

$$y_{i,f,j,g} = y_{j\ g\ i\ f} \geq 0 \quad \forall i\ j \in V \ \forall f\ \ g \in N \tag{8}$$

$$\sum_{i\in V} x_{i,f} = 1 \quad \forall f \in N \tag{9}$$

$$\sum_{f\in N} x_{i,f} = 1 \quad \forall i \in V \tag{10}$$

$$x_{i\ f} \in \{0,1\} \quad \forall i \in V \ \forall f \in N, \tag{11}$$

By relaxing the binary constraint (11) (using a LP relaxation), the above formulation can be used to obtain a valid lower bound on the QAP (1)-(4). The RLT-1 reformulation was further strengthened by introducing additional n^6 variables, called RLT-2 in (Adams et al., 2007), and even further with additional n^8 variables, called RLT-3 in (Hahn et al., 2012), which for the time being is still too large even for modern day computers – for problems of size $n = 25$, the computations needed to be done on a server with 384 GB of RAM. A different mixed integer linearization scheme, called the Kaufman-Broeckx formulation, was proposed in (Kaufman & Broeckx, 1978) with $O(n^2)$ additional variables. Although this is the smallest QAP linearization, its LP relaxation is known to be usually weak. This relaxation was tightened in (Xia & Yuan, 2006) using the Gilmore-Lawler bound (GLB) (Gilmore, 1962; Lawler, 1963) and further enhanced in (Zhang et al., 2013). A formulation based in the Kaufman-Broeckx family was used in (Fischetti et al., 2012) to solve (prove optimality) all the esc instances (Eschermann & Wunderlich, 1990) (including the one of size $n = 128$).

2.3. Lower Bounding Techniques

Exact solution of a QAP typically requires the use of a branch-and-bound framework (Anstreicher, 2003). In practice, the lack of efficiently computable, tight lower bounds for the QAP has been the key factor in the problem's difficulty, as the tighter the bound is, the more difficult it generally is to compute. There are various approaches for obtaining lower bounds. One of the oldest methods, the Gilmore-Lawler bound (GLB), is still widely used. A comparison of older bounds based on linearization of the QAP can be found in (Karisch et al., 1999). A great success in solving previously unsolved QAP instances was achieved using the convex quadratic programming bound introduced in (Anstreicher & Brixius, 2001).

A seminal breaking point in combinatorial optimization was the emergence of semidefinite programming (SDP). The SDP bounds for the QAP were first studied in (Zhao et al., 1998). The problem with this relaxation was that it involved a matrix variable of order n^2, and can therefore only be solved efficiently by interior point methods for, say, $n \leq 20$. This limitation has encouraged research into exploiting group symmetry of the QAP data matrices to obtain smaller and more tractable SDP problems (de Klerk & Sotirov, 2012). It has also prompted recent research into SDP relaxations of QAP where the matrix variables are of order n; see (Peng et al., 2010) and (Peng et al., 2015). In both these lines of research the authors were able to compute the best-known lower bounds for some QAPLIB instances. As we will use the lower bounding techniques for the construction of starting points for our metaheuristic, we will describe these techniques in greater detail. The computation of the GLB can be done in the following way: Denote the row vectors of matrices A and B by a_i and $b_i, i = 1,2,\dots,n,$. Let $\hat{a}_i$ be the vector consisting of the $(n-1)$ components of a_i, without $a_{i,i}$, and let $\hat{b}_i$ be the vector consisting of the $(n-1)$ components of b_i, without $b_{i,i}$. Define a matrix $L = (l_{i,j})$ as follows:

$$l_{ij} = \langle \hat{a}_i, \hat{b}_j \rangle_{-}, \quad i,j = 1,2,\dots,n,$$

where $\langle a, b \rangle_{-}$ is the minimal scalar product, which can be computed by ordering the vector a nondecreasingly and b nonincreasingly. The GLB is given by the optimal value of the n-dimensional LAP with cost matrix $l_{i,j} + a_{i,i}b_{j,j}$:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(l_{i,j} + a_{i,i}b_{j,j}\right)x_{i\ j} \tag{12}$$

$$\text{s.t.} \quad \sum_{i\in V} x_{i,f} = 1 \quad \forall f \in N \tag{13}$$

$$\sum_{f \in N} x_{i,f} = 1 \quad \forall i \in V \tag{14}$$

$$x_{i\,f} \in \{0,\ 1\} \quad \forall i \in V\ \forall f \in N \tag{15}$$

which requires only $O(n^3)$ computational time.

The second type of the QAP lower bounds are the eigenvalue bounds. These use the fact that the set of permutation matrices Π_n can be characterized as:

$$\Pi_n = \mathcal{Q}_n \cap \mathcal{E}_n \cap \mathcal{N}_n,$$

where $\mathcal{Q}_n$ is the set of orthogonal matrices, $\mathcal{E}_n$ is the set of doubly stochastic matrices and $\mathcal{N}_n$ is the set of matrices with positive elements of size $n \times n$.
The QAP can then be equivalently formulated as:

$$\min_{X \in \Pi_n} tr\,(AXBX^T),$$

where tr$(\cdot)$ is the trace of a matrix. The first eigenvalue bound that uses this QAP formulation was introduced in (Hoffman & Wielandt, 2003) and is based on the relaxation of the feasible region:

$$\min_{X \in \mathcal{Q}_n} \mathrm{tr}\,(AXBX^T) = \langle \lambda(A), \lambda(B) \rangle_-,$$

where $\lambda(\cdot)$ denotes the vector of eigenvalues of the matrix. This bound can be computed with very little effort but tends to be extremely weak. The improvement of this bound was done in (Hadley et al., 1992) and is called the Hadley-Rendl-Wolkowitz (HRW): Let u_n be a vector of all ones and let V be an $n \times (n-1)$ matrix with $V^T u_n = 0$ and rank$(V) = n - 1$. Then

$$\{X \in \mathcal{R}^{n \times n}: Xu_n = X^T u_n = u_n\} = \{\frac{1}{n} u_n u_n^T + VMV^T: M \in \mathcal{R}^{(n-1)\times(n-1)}\}$$

which can be used to reparametrize the trace formulation as:

$$\mathrm{tr}(AXBX^T) = tr\left((V^TAV)\hat{X}(V^TBV)\hat{X}^T\right) + \frac{2}{n}\mathrm{tr}(AJ_nB)X^T - \mathrm{const}$$

where $J_n = u_n u_n^T$ is a matrix of all ones, and use the eigenvalue bound to obtain the improved HRW bound:

$$\langle \lambda(V^TAV)\ \lambda(V^TBV) \rangle_- + \mathrm{LAP}\left(\frac{2}{n} AJ_nB\right) - \mathrm{const} \tag{16}$$

The third type of the QAP lower bounds are based on a convex quadratic programming relaxation of the trace reparameterization shown above. (Anstreicher & Brixius, 2001) used the above-mentioned parametrization and showed that the following convex quadratic optimization problem gives a lower bound on the QAP:

$$\min \quad \mathrm{vec}(X)^T Q \mathrm{vec}(X) + \langle \lambda(V^TAV)\ \lambda(V^TBV) \rangle_- \tag{17}$$

$$\text{s.t.} \quad Xu_n = X^T u_n = u_n \quad X \geq 0, \tag{18}$$

where

$$Q = (B \otimes A) - \left(I \otimes V\hat{S}V^T\right) - \left(V\hat{T}V^T \otimes I\right)$$

and $\hat{S}$ and $\hat{T}$ can be obtained from the spectral decomposition of V^TAV and V^TBV. The last type of the QAP lower bounds we consider are based on a SDP relaxation developed by (Peng et al., 2015): Let (B_1, B_2) be a minimal trace matrix splitting of the matrix B and compute a decomposition $B_i = \hat{B}_i^T \hat{B}_i$. Let $B_s = B_1 + B_2$ the SDP relaxation model of QAP based on minimal trace matrix splitting is the following:

$$\min \quad \mathrm{tr(AY)} \tag{19}$$

$$\text{s.t.} \quad Y = Y_1 - Y_2 \quad Y_s = Y_1 + Y_2 \tag{20}$$

$$\begin{pmatrix} I & \hat{B}_1 X^T \\ X\hat{B}_1 & Y_1 \end{pmatrix} \succeq 0 \begin{pmatrix} I & \hat{B}_2 X^T \\ X\hat{B}_2 & Y_2 \end{pmatrix} \succeq 0 \tag{21}$$

$$\mathrm{diag}(Y_1) = X\mathrm{diag}(B_1) \quad Y_1 e = XB_1 e \tag{22}$$

$$\mathrm{diag}(Y_2) = X\mathrm{diag}(B_2) \quad Y_2 e = XB_2 e \tag{23}$$

$$\left(X \min\left([B_1]_{off}\right)\right)_i \leq [Y_1]_{i\,j} \quad \forall i \neq j \tag{24}$$

$$[Y_1]_{i\,j} \leq \left(X \max\left([B_1]_{off}\right)\right)_i \quad \forall i \neq j \tag{25}$$

$$\left(X \min\left([B_2]_{off}\right)\right)_i \leq [Y_2]_{i\,j} \quad \forall i \neq j \tag{26}$$

$$[Y_2]_{i\,j} \leq \left(X \max\left([B_2]_{off}\right)\right)_i \quad \forall i \neq j \tag{27}$$

$$\left(X \min\left([B]_{off}\right)\right)_i \leq [Y]_{i\,j} \quad \forall i \neq j \tag{28}$$

$$[Y]_{i\,j} \leq \left(X \max\left([B]_{off}\right)\right)_i \quad \forall i \neq j \tag{29}$$

$$\left(X \min\left([B_s]_{off}\right)\right)_i \leq [Y_s]_{i\,j} \quad \forall i \neq j \tag{30}$$

$$[Y_s]_{i\,j} \leq \left(X \max\left([B_s]_{off}\right)\right)_i \quad \forall i \neq j \tag{31}$$

$$\|[Y_1]_{i\,:}\|_2 \leq X\, \|[B_1]_{i\,:}\|_2 \quad \forall i \tag{32}$$

$$\|[Y_2]_{i\,:}\|_2 \leq X\, \|[B_2]_{i\,:}\|_2 \quad \forall i \tag{33}$$

$$\|[Y]_{i\,:}\|_2 \leq X\, \|[B]_{i\,:}\|_2 \quad \forall i \tag{34}$$

$$\|[Y_s]_{i\,:}\|_2 \leq X\, \|[B_s]_{i\,:}\|_2 \quad \forall i \tag{35}$$

$$X \geq 0 \;\; Xe = X^T e = e \tag{36}$$

B_{off} denotes the matrix consisting of all the off-diagonal elements of B, i.e., $B_{off} = B - \text{diag}(b_{11}, b_{22}, \dots, b_{nn})$, $\max(B)$ (or $\min(B)$)denotes the vector whose i-th component is the maximal element (or minimal element) in the i-th row (denoted by $B_{i,:}$) of B.

In order to obtain a starting point (in our case, a starting permutation) for the upcoming heuristic, we use a projection of the matrix obtained from the lower bounding schemes to the space of permutation matrices: Let $\hat{X}$ be a matrix obtained from the computation of the lower bounds. The "closest" permutation matrix X to $\hat{X}$ can be computed by solving the following problem:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(X_{i,j} - \hat{X}_{i,j}\right)^2 \tag{37}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} X_{i,\,j} = 1 \;\sum_{i=1}^{n} X_{i,\,j} = 1 \quad \forall i \;\forall j \tag{38}$$

$$X_{i\,j} \in \{0\ ,1\} \quad \forall i \;\forall j, \tag{39}$$

The problem above can be reformulated into an equivalent problem using the fact that $X_{i,j}$ is binary:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(1 - 2\hat{X}_{i,\,j}\right)X_{i,\,j} \tag{40}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} X_{i,\,j} = 1 \;\sum_{i=1}^{n} X_{i,j} = 1 \quad \forall i \;\forall j \tag{41}$$

$$X_{i\,j} \in \{0,1\} \quad \forall i \;\forall j, \tag{42}$$

which is a simple LAP.

2.4 HC12 algorithm

The binary HC12 algorithm, described in detail in (Matousek & Zampachova, 2011), is a stochastic heuristic searching algorithm which belongs to the class of pseudo global search methods. The basic step of the algorithm is a generation of a neighborhood of the current solution, which serves as a base for the construction of a new (improved) population. The method of generating the neighborhood is the pivotal characteristic of HC12. The paradigm of the algorithm is the search of the optimal solution in the binary (Hamming) space, that encodes the solution. In this context, it is a parallel approach to genetic algorithms, where the solution is encoded as a binary vector. The best individual of the ith generation (or iteration) is chosen as the base for the following $(i + 1)$ generation. The approach is depicted in Fig. 1. The binary vector of the current solution is called a kernel a is denoted with an index "ker" (e.g., a_{ker}). The newly generated neighborhood creates a set of c new binary vectors a_i with the same length as the vector a_{ker}. These new vectors can be viewed as a population and represented by a matrix $A_0 = (a_1, \dots, a_c)^T$. The degree of locality/globality of the optimization depends on the particular way the new population $A_0$is generated. The goal of the search is to find optimal parameters x_{opt}(43) with respect to the define objective function $f(x)$ on a parametric space $D \in N$. Because of the binary representation, the parametric space is defined by a mapping Γ: $\{0,1\} \rightarrow D$. An important implementation detail of the mapping Γ is the translation of the binary vector from the Gray code into direct binary; afterwards, there is a problem-based decoding of the binary vector (0-1 problem, integer problem, or mixed integer problem). The following relationship is used $x = \Gamma(a)$ to denote the optimal solution as follows:

$$x_{opt} = \text{argmin}_{x \in D} \text{f}(x) \tag{43}$$

$$a_{opt} = \text{argmin}_{a \in \{0\ 1\}^n} f\big(\Gamma(a)\big) \tag{44}$$

Over this binary representation is defined the neighborhood relation, that describes the neighborhood s for each feasible a_{ker} as points $a \in s(a_{ker})$. The choice of the neighborhood function s determines the behavior and character of the HC algorithm (Fig. 1).

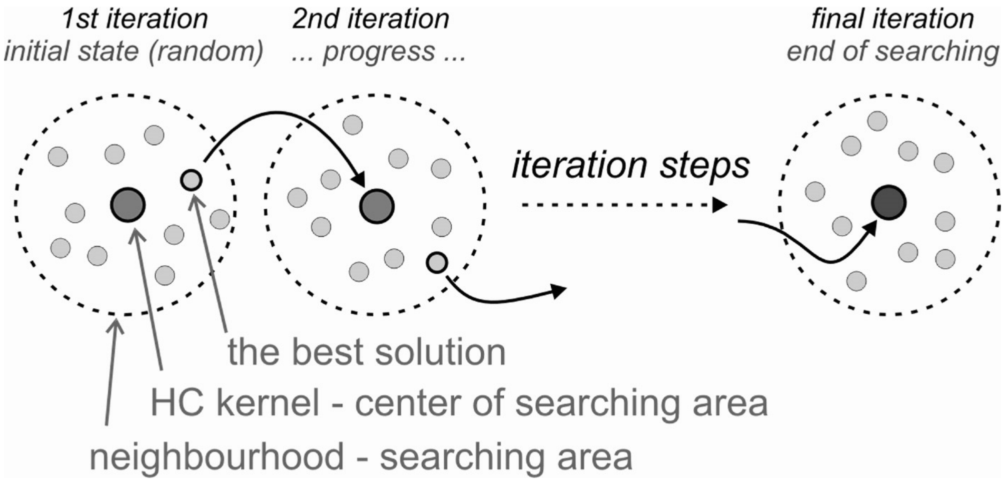

Fig. 1. An schematic example of the progress of the binary HC algorithm

The HC12 algorithm is very effectively parallelizable. Using the neighborhood function s (45) on a binary vector a_{ker}, the population A_0 is generated. The set of possible neighborhood functions is denoted by H (46).

$$s: a_{ker} \to A_0 \quad \text{i, e,}\ \ s: \{0\ 1\}^n \to (\{0\ 1\}^n)^c \tag{45}$$

$$H = \{s_0\ \ s_1\ \ldots\ s_n\} \tag{46}$$

The number c of newly generated vectors in the population A_0 depends on the chosen neighborhood function s_k and on the length n of the binary vector a_{ker} – it is computed as $c = \binom{n}{k}$. For the realization of the transformations from the set H a system of matrices M is defined. The matrix M corresponding to the function s_k will be called a matrix of the k-th order and denoted by M_k. Matrix of the k-th order (M_k) is a matrix whose rows represent all points of the Hamming metric space that are distance k from the origin (zero vector of length n):

$$M_0 = (0_{1,1} \quad 0_{1,2} \quad \cdots \quad 0_{1,n})$$

$$M_1 = \begin{pmatrix} 1_{1,1} & 0_{1,2} & 0_{1,3} & \cdots & 0_{1,n} \\ 0_{2,1} & 1_{2,2} & 0_{2,3} & \cdots & 0_{2,n} \\ \vdots & & & \ddots & \\ 0_{c_1,1} & 0_{c_1,2} & 0_{c_1,3} & \cdots & 1_{c_1,n} \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 1_{1,1} & 1_{1,2} & 0_{1,3} & \cdots & 0_{1,n} \\ 1_{2,1} & 0_{2,2} & 1_{2,3} & \cdots & 0_{2,n} \\ \vdots & & & \ddots & \\ 0_{c_2,1} & 0_{c_2,2} & \cdots & 1_{c_2,n-1} & 1_{c_2,n} \end{pmatrix}$$

$$\vdots$$

$$M_n = (1_{1,1} \quad 1_{1,2} \quad \cdots \quad 1_{1,n})$$

Using the k-th order matrices, the function s_k can be effectively computed

as:

$$s_k\colon A_0 = a_{ker} \oplus M_k$$

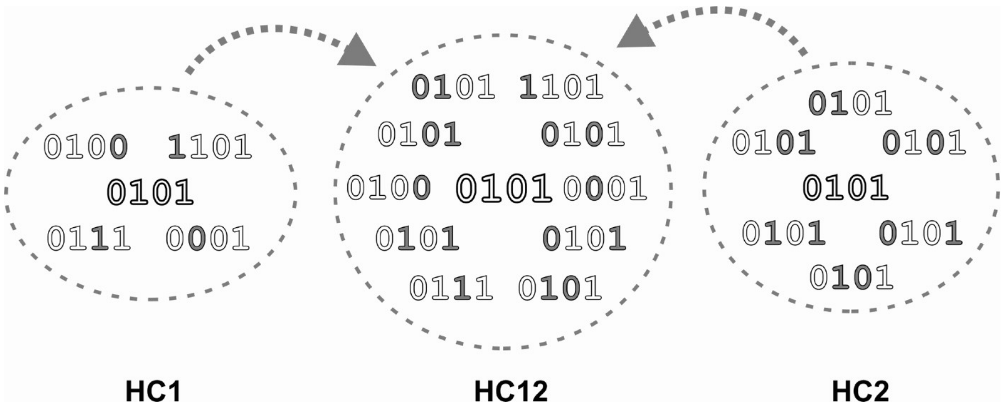

Fig. 2. An example of neighbourhood generation for 4-bit binary string using transformations $H = \{s_0, s_1, s_2\}$ and matrixes M_0, M_1, M_2, i.e. utilization in algorithms HC1, HC2 and their union HC12

From the practical point of view (because of the combinatorial expansion), only the transformations M_0, M_1, and M_2 are used. The algorithm HC12 encodes in the last digit of its name the utilized transformations (upto order 2). The general paradigm of the HC12 algorithm is implemented using several input parameters: *fun* (indicator of the objective function f), *nRun* (the number of runs/restarts of the algorithm). The value *nRun* depends on the difficulty of the problem. The section of rows 6 to 10 are the HC12 algorithm itself. This part is very well (row 8) and well (row 9) parallelizable. The computations in row 8 contain the conversion from the Gray code into direct binary, implicitly. The main focus of this paper is on row 5 of the algorithm. How does one select a good starting solution? One possibility is to start at a random solution with the hope that after a sufficiently large number of tries, one does get a "good enough" solution. The other possibility is to start from a solution that is obtained by some heuristic. In this case, the heuristic in question entails the computation of the lower bound for the QAP (by one of the methods described earlier) and the projection of the obtained lower bound solution on the space of permutation matrices, by solving (40)-(42). The implementation of the HC12 algorithm for the QAP was described in (Matousek et al., 2019). As noted earlier, the optimization problem in (1)-(4) can be interpreted as a search over the space of permutation matrices $X \in \Pi_n$. From the problem structure of the QAP it is clear that swapping arbitrary columns of a (feasible) matrix X always results in a different feasible matrix (and swapping the rows of the matrix has the same effect).

Algorithm 1 The HC12 algorithm (Pseudo code of the general paradigm).

```
1:   *fun,nRun* ← **inputs**
2:   $M \leftarrow (M_0, M_1, M_2)^T$
3:   $f_{best} \leftarrow \infty$
4:   **for** $i = [1 : nRun]$ **do**
5:       $a_{opt} \leftarrow$ **random / heuristic**
6:       **repeat**
7:           $a_{ker} \leftarrow a_{opt}$
8:           $A \leftarrow a_{ker} \oplus M$
9:           $a_{opt} \leftarrow \mathrm{argmin}_{a\in\{0,1\}^n} f(\Gamma(a))$
10:      **until** $a_{opt} = a_{ker}$
11:      $f_{best}(i) \leftarrow f\left(\Gamma(a_{opt})\right)$
12:      $A_{best}(i,:) \leftarrow a_{opt}$
13:  **endfor**
14:  $[i, f_{min}] \leftarrow \min_i f_{best}(i)$
15:  $a_{min} \leftarrow A_{best}(i,:)$
16:  $x_{min} \leftarrow \Gamma(a_{min})$
17:  **return** $\{f_{min}, a_{min}, x_{min}\}$
```

4. Results and discussion

The computational experiments were carried out on 53 symmetrical QAP instances from the QAPLIB. For these instances the GLB (12)-(15), HRW (16), convex quadratic (17)-(18), and semidefinite (19)-(36) bounds, and their projections (40)-(42) were computed. For the computation of the convex quadratic (AB) and semidefinite (PE) bounds, and for the computation of the LAP for the projection, the corresponding optimization problems were implemented in JuMP environment in JULIA language and the MOSEK solver was used to obtain the solutions. The results from these computations are summarized in Table 1. For one of the instances (tho150), the AB and PE formulations were too big to handle. These computations were carried out on Intel Xeon E5530 2.40GHz CPU with 16GB of RAM. The HC12 algorithm was implemented for HPC computations on GPU CUDA 7.x (i.e., NVIDIA RTX 2080, 8GB), where not more than 6GB were used for any of the QAP instances. From Table 1 we can see that, at least in general, the more complicated formulations (convex quadratic and semidefinite) produce better (higher) lower bounds, but not necessarily better (lower) starting point values, when judged solely on the resulting projection. The trade-off is that these more complicated formulations need quite a lot more computational resources (judged by the computational time) and are only feasible for instances up to $n = 100$. Also, every method produced the best lower bound and best projected value for at least one problem instance. It should be noted that the lower bounds are not only useful for constructing possible starting solutions for heuristics, but also help to judge the closeness of the solution obtained by the heuristic to the true optimum. This is especially important in situation, where the is otherwise no information about what the optimal value of the QAP instance might be. Next, we used the projected values from the lower bounding techniques as the starting points for the HC12 metaheuristic and run it 1,000 times for each problem instance. The best results of these simulations (the solution with the lowest objective value out of the 1,000) are reported in Table 2. We also include the results from simulations that used random permutations as the starting point (again 1,000). Similarly, to the results of the lower bounds, there is not a clear winner, as for each of the methods (even for the random start) there are instances where it produced solutions that were better than the ones from the other methods. However, we can compare each of the bounding methods with the random start to see if there is significant difference. This comparison is summarized in Table 3 – we can see that even the "worst" performing lower bounding technique (HWB) was significantly better that random start, beating it in 30 of the 53 instances. The "best" performing lower bounding technique was the most complicated semidefinite formulation (PE), which was better than random start in 41 of the 52 instances. We can also see that the GLB method performed a bit better than the much more complicated convex quadratic (AB) one. Similar pattern can be observed for the median results reported in Table 4. The main difference is that the random start was never the best scoring method, while each of the lower bounding methods were the best in at least 6 instances. The comparison of the lower bounding method with random start for median results summarized in Table 5 shows even bigger difference than the one for best (minimum) results – the "worst" lower bounding technique (HWB, again) was better than random start in 41 of the 53 instances, and the "best" one (PE, again) was better in every one of the 52 instances. The GLB and AB methods perform similarly well. From these results, it is clear that starting a heuristic from a carefully chosen points leads to an increase in quality of the resulting solutions. The choice of the technique for constructing these starting points mainly depends on the computational resources at our disposal. While for the QAP the semidefinite (PE) formulation produced the best behaving starting points, it was also the most computationally demanding method, requiring the use of advanced convex optimization algorithms or the use of powerful solvers. In contrast to this, the GLB method produce starting points that are almost as good as the PE one, but the computational requirements for GLB are negligible.

5. Conclusion

In this paper we have studied the effects of using the lower bounding techniques for the QAP as for the generation of starting points for the HC12 heuristic, that subsequently tried to find the optimal solution for the QAP. We have shown through extensive numerical computations that this utilization of the lower bounding techniques significantly improves the values of the resulting solutions. Out of the four compared lower bounding techniques, the best overall results were obtained by using the semidefinite relaxation method, which was also the most computationally demanding one. When the computational resources, or the access to high quality semidefinite optimization solvers are limited, the GLB bound can serve as an excellent surrogate – although the resulting solutions are not as good, the computational requirements are negligible.

Future research will focus on extending the multicriteria and stochastic QAP instances. Also, the evaluation of various other heuristics that can use the starting points could be interesting, as different methods could benefit more (or less) from starting from an already decent point. Lastly, we expect to work on the evaluation of the starting solutions for other NP-hard optimization problems.

Table 1
Lower bounds (LB), values of the projections (PV), and computational time (T) of the four considered QAP lower bounding techniques for selected QAPLIB problems (BKW – best known value). Best results (highest for LB and lowest for PV) are emphasized in bold.

		GLB (Gilmore, 1962)			HRW (Hadley et al., 1992)			AB (Anstreicher & Brixius, 2001)			PE (Peng et al., 2015)		
Instance	BKW	LB	PV	T [s]	LB	PV	T [s]	LB	PV	T [s]	LB	PV	T [s]
chr12a	9552	724	44232	0.001	0	25638	0.001	0	**23246**	0.095	**8499**	25752	1.136
chr12b	9742	7146	**25580**	0.001	0	26712	0.001	0	27088	0.107	**7340**	41170	1.317
chr12c	11156	**7976**	**18784**	0.001	0	31608	0.001	0	30876	0.089	9832	40438	1.052
chr15a	9896	5625	50174	0.001	0	25958	0.001	0	**25880**	0.253	**7441**	46976	2.515
chr15b	7990	4653	54254	0.001	0	38470	0.001	0	**19008**	0.256	**5166**	30798	2.870
chr15c	9504	6165	44602	0.001	0	**26144**	0.001	0	35936	0.260	**8808**	43370	2.207
chr18a	11098	6779	89486	0.001	0	**38778**	0.001	0	51800	0.331	**9077**	78282	4.383
chr18b	1534	**1534**	4640	0.001	0	**2372**	0.001	0	3922	0.338	**1534**	4156	2.580
chr20a	2192	2150	9778	0.001	0	7742	0.001	0	**6904**	0.489	**2156**	10302	5.814
chr20b	2298	2196	10430	0.001	0	**6418**	0.001	0	7386	0.497	**2237**	10320	7.654
chr20c	14142	8601	79430	0.013	0	**63268**	0.001	0	63350	0.511	**8825**	69124	7.588
chr22a	6156	5924	17622	0.003	0	**10178**	0.001	0	10588	0.815	**5964**	10316	8.907
chr22b	6194	5936	13486	0.002	0	**9530**	0.001	0	11744	0.995	**6015**	10264	7.664
chr25a	3796	2765	18964	0.002	0	14186	0.001	0	13062	1.444	**3244**	**11708**	17.67
had20	6922	6166	7550	0.001	6625	7460	0.001	6671	**7184**	0.519	**6778**	7486	4.486
kra30a	88900	68360	120000	0.004	63717	**117490**	0.001	68467	111970	10.52	**73983**	118920	34.27
kra30b	91420	69065	**118720**	0.005	120990	140040	0.001	68876	122440	3.782	**68737**	127900	20.88
kra32	88700	67390	121620	0.007	59735	119450	0.002	64591	**117010**	4.780	**72297**	119540	29.96
nug18	1930	1554	2402	0.002	1663	**2250**	0.001	1703	2320	0.303	**1753**	2278	2.937
nug20	2570	2057	3080	0.002	2196	2940	0.001	2253	**2884**	0.481	**2338**	2972	4.191
nug21	2438	1833	3198	0.002	1979	**2914**	0.001	2051	3030	0.627	**2215**	3266	5.777
nug22	3596	2483	4598	0.002	2966	4448	0.001	3074	4346	0.816	**3284**	**4312**	9.638
nug24	3488	2676	4222	0.035	2960	4170	0.001	3024	4272	1.378	**3178**	**4066**	7.517
nug25	3744	2869	4728	0.003	3190	4634	0.001	3267	4534	1.521	**3404**	**4496**	8.902
nug27	5234	3701	**6246**	0.003	4493	6586	0.001	4604	6378	2.495	**4820**	6548	18.64
nug28	5166	3786	6470	0.003	4433	6310	0.001	4538	**6208**	2.705	**4732**	6274	15.07
nug30	6124	4539	7706	0.003	5266	7342	0.001	5360	**7270**	3.702	**5608**	7664	25.51
scr15	51140	44737	**70154**	0.002	10355	75950	0.001	12478	77786	0.299	**46015**	83280	1.339
scr20	110030	86766	203736	0.001	16113	**172306**	0.001	22714	193250	0.474	**92426**	193396	5.518
sko42	15812	11311	19522	0.008	13830	19088	0.002	14029	18748	18.23	**14612**	**18492**	121.7
sko64	48498	32522	57316	0.030	43890	56376	0.003	44513	**56214**	126.2	**45467**	56766	623.7
sko72	66256	44280	75860	0.048	60402	75772	0.004	61069	**75632**	222.4	**61497**	76082	1209
sko81	90998	60283	105932	0.061	82277	104844	0.005	83433	**103642**	466.3	**85795**	103954	2757
sko90	115534	75531	132996	0.083	105983	**131398**	0.006	107171	131794	1448	**109260**	131646	4585
sko100a	152002	98953	171886	0.091	139365	**170880**	0.009	140946	172554	2847	**144091**	171294	9946
sko100b	153890	99028	174226	0.100	141251	**173878**	0.007	143138	174290	3055	**145783**	174950	10745
sko100c	147862	95979	169888	0.101	135011	**167142**	0.007	136773	170340	3306	**140146**	169792	8920
sko100d	149576	95921	172024	0.091	136979	**167642**	0.007	138736	169774	2928	**140072**	171152	10162
sko100e	149150	95551	171360	0.093	136996	168732	0.007	138711	**168640**	3090	**139277**	169914	9688
sko100f	149036	96016	169768	0.093	136860	170008	0.007	138661	**169090**	3634	**140885**	169564	10627
ste36a	9526	7124	**14866**	0.006	0	16112	0.001	0	17454	9.257	**7731**	20690	60.95
ste36b	15852	8653	47768	0.007	0	42034	0.001	0	**37398**	8.375	**12930**	51936	60.06
ste36c	8.239e6	6.393e6	2.191e7	0.006	0	2.119e7	0.001	0	1.952e7	8.193	**6.546e6**	**1.775e7**	101.4
tai25a	1.167e6	962417	1.457e6	0.002	956657	**1.310e6**	0.001	**967207**	1.366e6	1.758	958027	1.411e6	12.89
tai50a	4.9386e	3.854e6	5.838e6	0.017	3.840e6	5.680e6	0.002	**3.870e6**	**5.613e6**	67.49	3.842e6	5.667e6	253.0
tai60a	7.205e6	5.55e6	8.481e6	0.27	5.537e6	8.398e6	0.004	**5.575e6**	8.464e6	82.67	5.544e6	**8.216e6**	841.2
tai80a	1.349e7	1.032e7	1.570e7	0.065	1.030e7	1.568e7	0.005	**1.035e7**	**1.573e7**	413.5	1.031e7	1.556e7	3213
tai100a	2.105e7	1.582e7	2.403e7	0.101	1.579e7	2.348e7	0.007	**1.585e7**	**2.347e7**	1667	1.552e7	2.347e7	6460
tho30	149936	90578	195698	0.003	119255	**193756**	0.001	124217	200194	4.067	**131588**	198154	27.20
tho40	240516	143804	**298906**	0.007	191042	303704	0.002	197661	319004	13.45	**210210**	313114	109.9
tho150	8.133e6	4.123e6	**9.703e6**	0.226	**7.350e6**	9.756e6	0.013	–	–	–	–	–	–
wil50	48816	38069	53942	0.014	45731	53420	0.003	46194	**52938**	40.64	**46901**	53754	281.4
wil100	273038	210949	293908	0.095	260827	293206	0.007	262584	**291630**	2591	**264724**	294948	11078

Table 2
Best (minimum) results from the simulations. If the BKW is confirmed optimal, it is highlighted in bold. Also, in bold is the method that produced the best solution for the given instance.

Instance	BKW	Rand	GLB	HRW	AB	PE
chr12a	**9552**	**9552**	**9552**	**9552**	**9552**	**9552**
chr12b	**9742**	**9742**	**9742**	**9742**	**9742**	**9742**
chr12c	**11156**	11186	**11156**	**11156**	**11156**	**11156**
chr15a	**9896**	10094	10010	9980	10106	**9978**
chr15b	**7990**	8626	**8210**	9096	8458	8452
chr15c	**9504**	10118	**9504**	10426	9940	10002
chr18a	**11098**	**11682**	**11682**	12396	12004	12424
chr18b	**1534**	1538	**1534**	**1534**	1538	**1534**
chr20a	**2192**	2480	2532	2592	**2398**	2402
chr20b	**2298**	2612	2608	**2598**	2674	2618
chr20c	**14142**	**14610**	14988	15636	17274	14876
chr22a	**6156**	6408	6342	6354	6456	**6336**
chr22b	**6194**	6522	6526	6534	6410	**6352**
chr25a	**3796**	5062	4678	5056	4970	**4230**
had20	**6922**	6924	6928	**6922**	6956	**6922**
kra30a	**88900**	93460	92480	93850	93460	**92300**
kra30b	**91420**	95020	94570	94690	93620	**92380**
kra32	**88700**	**91660**	92420	92270	92650	92320
nug18	**1930**	1958	**1936**	1950	1938	1938
nug20	**2570**	2598	2614	**2590**	2602	2598
nug21	**2438**	2458	2452	2472	**2450**	2452
nug22	**3596**	3628	**3610**	3628	3628	**3610**
nug24	**3488**	3552	3554	3582	3546	**3528**
nug25	**3744**	3806	3788	3800	**3760**	3762
nug27	**5234**	5298	5298	5328	**5294**	5304
nug28	**5166**	5314	5284	5288	5272	**5260**
nug30	**6124**	6272	6260	6316	6254	**6220**
scr15	**51140**	**51140**	52340	**51140**	**51140**	**51140**
scr20	**110030**	111078	111938	110802	112660	**110772**
sko42	15812	16304	16282	16290	**16106**	16172
sko64	48498	50090	**49904**	49932	49970	49942
sko72	66256	68298	68182	68264	**67902**	68140
sko81	90998	93684	93492	93968	93840	**93148**
sko90	115534	119630	119064	119078	119092	**118446**
sko100a	152002	157426	157034	156934	**156116**	156820
sko100b	153890	159060	**158002**	158456	158220	158184
sko100c	147862	152742	152592	153186	152476	**152374**
sko100d	149576	154708	154340	**153896**	153978	154144
sko100e	149150	**153880**	154522	153930	154010	154536
sko100f	149036	154284	153994	153788	153766	**153558**
ste36a	**9526**	10234	10126	10052	**9790**	10266
ste36b	**15852**	17786	17140	17112	**16724**	17770
ste36c	**8239110**	8701576	8678652	8738822	8695554	**8578694**
tai25a	**1167256**	1194194	**1177180**	1188890	1188248	1187984
tai50a	4938796	5148702	**5119448**	5131652	5130768	5132594
tai60a	7205962	7486562	7506384	7499212	7434242	**7427410**
tai80a	13499184	14034018	14027470	**13966388**	14050896	13993668
tai100a	21052466	21951138	21932812	21957694	**21893240**	21932070
tho30	**149936**	154134	156170	153558	154874	**152020**
tho40	240516	251428	249370	248116	**247260**	251034
tho150	8133398	8511942	8461186	**8454432**	–	–
wil50	48816	49504	49472	49652	**49434**	49470
wil100	273038	278428	**277210**	277730	277230	277458

Table 3
Comparison of the lower bounding methods with Rand – best (minimum) results.

	GLB	HWB	AB	PE
Rand better	12	19	13	7
Rand worst	37	30	33	41
Rand equal	4	4	6	4

Table 4
Median results from the simulations. If the BKW is confirmed optimal, it is highlighted in bold. Also, in bold is the method that produced the lowest median value for the given instance.

Instance	BKW	Rand	GLB	HRW	AB	PE
chr12a	**9552**	12531	11866	**11550**	12798	12130
chr12b	**9742**	13170	**10570**	11628	11548	11886
chr12c	**11156**	14221	**13060**	14139	13518	13355
chr15a	**9896**	14691	13886	13850	**13625**	14264
chr15b	**7990**	13505	12334	13637	12490	**12142**
chr15c	**9504**	15445	**14303**	15812	14518	14675
chr18a	**11098**	19074	18159	18664	17565	**17237**
chr18b	**1534**	1779	**1730**	1760	1776	1734
chr20a	**2192**	3480	3374	3406	**3280**	3359
chr20b	**2298**	3455	3384	3468	3453	**3272**
chr20c	**14142**	25957	**23068**	24678	27073	23338
chr22a	**6156**	7168	7007	7016	7144	**6962**
chr22b	**6194**	7218	7086	7227	7078	**7059**
chr25a	**3796**	6819	6068	6677	6520	**5687**
had20	**6922**	7008	7002	7018	**6982**	7000
kra30a	**88900**	99285	98410	100070	97930	**97340**
kra30b	**91420**	100360	100945	100275	100695	**100145**
kra32	**88700**	98365	98360	98435	98340	**98260**
nug18	**1930**	2038	2022	**2010**	2018	2012
nug20	**2570**	2728	2708	2708	**2704**	2714
nug21	**2438**	2596	2562	2570	**2548**	2560
nug22	**3596**	3789	3738	3878	3740	**3734**
nug24	**3488**	3754	3731	3744	3689	**3670**
nug25	**3744**	4002	3950	3930	**3920**	3940
nug27	**5234**	5588	5544	5532	5497	**5486**
nug28	**5166**	5546	5492	5492	5462	**5460**
nug30	**6124**	6562	6490	6524	6518	**6474**
scr15	**51140**	57428	56850	**56240**	56604	56613
scr20	**110030**	125945	124260	122999	**121845**	122096
sko42	15812	16854	16728	16830	**16661**	16663
sko64	48498	51228	50996	51092	50894	**50853**
sko72	66256	69782	69362	69625	69278	**69267**
sko81	90998	95591	95186	95379	95093	**94751**
sko90	115534	121745	121200	**120730**	121122	120980
sko100a	152002	159834	159447	159059	**158770**	159425
sko100b	153890	161739	160704	160686	**160100**	160450
sko100c	147862	156070	155364	155557	**154906**	155008
sko100d	149576	157353	156497	156886	**156012**	156377
sko100e	149150	157467	156798	156325	156646	**156178**
sko100f	149036	156510	155907	**155665**	155841	155841
ste36a	**9526**	11375	11134	11126	**11062**	11192
ste36b	**15852**	21794	20978	**20601**	20783	20899
ste36c	**8239110**	9523806	9564108	9602558	9566412	**9411506**
tai25a	**1167256**	1227125	**1223279**	1226160	1223803	1226389
tai50a	4938796	5266951	5249145	5255063	5235616	**5234108**
tai60a	7205962	7629342	7632746	7622290	7617218	**7592056**
tai80a	13499184	14235086	14239312	14251040	14249678	**14214487**
tai100a	21052466	22265769	22209784	22277072	**22202771**	22230955
tho30	**149936**	162522	163144	162024	161934	**159821**
tho40	240516	262386	259343	259822	**258655**	262188
tho150	8133398	8647360	8609647	**8594449**	–	–
wil50	48816	50434	50269	50544	**50092**	50126
wil100	273038	280670	280037	279836	**279480**	279705

Table 5
Comparison of the lower bounding methods with Rand – median results

	GLB	HWB	AB	PE
Rand better	5	12	5	0
Rand worst	47	41	47	52

Acknowledgement

This work was supported by IGA BUT: FSI-S-20-6538.

References

Abdel-Basset, M., Manogaran, G., El-Shahat, D., & Mirjalili, S. (2018). Integrating the whale algorithm with Tabu search for quadratic assignment problem: A new approach for locating hospital departments. *Applied soft computing*, 73, 530-546.

Adams, W. P., & Johnson, T. A. (1994). Improved linear programming-based lower bounds for the quadratic assignment problem. *DIMACS series in discrete mathematics and theoretical computer science*, 16, 43-77.

Adams, W. P., Guignard, M., Hahn, P. M., & Hightower, W. L. (2007). A level-2 reformulation–linearization technique bound for the quadratic assignment problem. European Journal of Operational Research, 180(3), 983-996.

Ahmed, Z. H. (2015). A multi-parent genetic algorithm for the quadratic assignment problem. *Opsearch*, 52(4), 714-732.

Aksan, Y., Dokeroglu, T., & Cosar, A. (2017). A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering*, 103, 105-115.

Anstreicher, K. M., & Brixius, N. W. (2001). A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming*, 89(3), 341-357.

Anstreicher, K. M. (2003). Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97(1), 27-42.

Benlic, U., & Hao, J. K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1), 584-595.

Burkard, R. E., & Rendl, F. (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17(2), 169-174.

Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). QAPLIB–a quadratic assignment problem library. *Journal of Global optimization*, 10(4), 391-403.

Burkard, R. E., Dell'Amico, M., & Martello, S. (2012). *Assignment problems: revised reprint*. Society for Industrial and Applied Mathematics.

Cela, E. (2013). The quadratic assignment problem: theory and algorithms (Vol. 1). Springer Science & Business Media.

Cela, E., Deineko, V., & Woeginger, G. J. (2018). New special cases of the Quadratic Assignment Problem with diagonally structured coefficient matrices. *European journal of operational research*, 267(3), 818-834.

Chmiel, W., & Szwed, P. (2016). Bees algorithm for the quadratic assignment problem on CUDA platform. In *Man–Machine Interactions 4* (pp. 615-625). Springer, Cham.

Czapiński, M. (2013). An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform. *Journal of Parallel and Distributed Computing*, 73(11), 1461-1468.

de Klerk, E., & Sotirov, R. (2012). Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. *Mathematical programming*, 133(1), 75-91.

Dell'Amico, M., Diaz, J. C. D., Iori, M., & Montanari, R. (2009). The single-finger keyboard layout problem. *Computers & Operations Research*, 36(11), 3002-3012.

Dickey, J. W., & Hopkins, J. W. (1972). Campus building arrangement using TOPAZ. *Transportation Research*, 6(1), 59-68.

Dokeroglu, T. (2015). Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem. *Computers & Industrial Engineering*, 85, 86-101.

Dokeroglu, T., & Cosar, A. (2016). A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, 52, 10-25.

Eschermann, B., & Wunderlich, H. J. (1990). Optimized synthesis of self-testable finite state machines. In *20th international symposium on fault-tolerant computing (FTCS 20).*

Fischetti, M., Monaci, M., & Salvagnin, D. (2012). Three ideas for the quadratic assignment problem. *Operations research*, 60(4), 954-964.

Fleurent, C., & Ferland, J. A. (1994). Genetic hybrids for the quadratic assignment problem. *Quadratic assignment and related problems*, 16, 173-187.

Gilmore, P. C. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the society for industrial and applied mathematics*, 10(2), 305-313.

Hadley, S. W., Rendl, F., & Wolkowicz, H. (1992). A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, 17(3), 727-739.

Hafiz, F., & Abdennour, A. (2016). Particle Swarm Algorithm variants for the Quadratic Assignment Problems-A probabilistic learning approach. *Expert Systems with Applications*, 44, 413-431.

Haghani, A., & Chen, M. C. (1998). Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6), 437-454.

Hahn, P. M., Zhu, Y. R., Guignard, M., Hightower, W. L., & Saltzman, M. J. (2012). A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. *INFORMS Journal on Computing*, 24(2), 202-209.

Hameed, A., Aboobaider, B., Mutar, M., & Choon, N. (2020). A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem. *International Journal of Industrial Engineering Computations*, 11(1), 51-72.

Harris, M., Berretta, R., Inostroza-Ponta, M., & Moscato, P. (2015, May). A memetic algorithm for the quadratic assignment problem with parallel local search. In *2015 IEEE congress on evolutionary computation (CEC)* (pp. 838-845). IEEE.

Helber, S., Böhme, D., Oucherif, F., Lagershausen, S., & Kasper, S. (2016). A hierarchical facility layout planning approach for large and complex hospitals. *Flexible services and manufacturing journal*, 28(1), 5-29.

Hoffman, A. J., & Wielandt, H. W. (2003). The variation of the spectrum of a normal matrix. In *Selected Papers Of Alan J Hoffman: With Commentary* (pp. 118-120).

Hubert, L., & Schultz, J. (1976). Quadratic assignment as a general data analysis strategy. *British journal of mathematical and statistical psychology*, 29(2), 190-241.

Hussin, M. S., & Stützle, T. (2014). Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. *Computers & operations research*, 43, 286-291.

Kaufman, L., & Broeckx, F. (1978). An algorithm for the quadratic assignment problem using Bender's decomposition. *European Journal of Operational Research*, 2(3), 207-211.

Karisch, S. E., Cela, E., Clausen, J., & Espersen, T. (1999). A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing*, 63(4), 351-403.

Koopmans, T. C., & Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, 53-76.

Matousek, R., & Zampachova, E. (2011). Promising GAHC and HC12 algorithms in global optimization tasks. *Optimization methods and software*, 26(3), 405-419.

Matousek, R., Popela, P., & Kudela, J. (2017). Heuristic approaches to stochastic quadratic assignment problem: Var and cvar cases. *Mendel,* 23(1), 73-78.

Matousek, R., Dobrovsky, L., & Kudela, J. (2019, July). The quadratic assignment problem: metaheuristic optimization using HC12 algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 153-154).

Mohammadi, J., Mirzaie, K., & Derhami, V. (2015, November). Parallel genetic algorithm based on GPU for solving quadratic assignment problem. In *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)* (pp. 569-572). IEEE.

Laporte, G., & Mercure, H. (1988). Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35(3), 378-381.

Lawler, E. L. (1963). The quadratic assignment problem. *Management science*, 9(4), 586-599.

Li, Y., & Pardalos, P. M. (1992). Generating quadratic assignment test problems with known optimal permutations. *Computational Optimization and Applications*, 1(2), 163-184.

Peng, J., Mittelmann, H., & Li, X. (2010). A new relaxation framework for quadratic assignment problems based on matrix splitting. *Mathematical Programming Computation*, 2(1), 59-77.

Peng, J., Zhu, T., Luo, H., & Toh, K. C. (2015). Semi-definite programming relaxation of quadratic assignment problems based on nonredundant matrix splitting. *Computational Optimization and Applications*, 60(1), 171-198.

Popela, P., Matousek, R., & Kudela, J. (2016). Heuristic approaches to stochastic quadratic assignment problem: VO and MM cases. *Mendel* 22(1), 117-122.

Samanta, S., Philip, D., & Chakraborty, S. (2018). Bi-objective dependent location quadratic assignment problem: Formulation and solution using a modified artificial bee colony algorithm. *Computers & Industrial Engineering*, 121, 8-26.

Sanhueza, C., Jiménez, F., Berretta, R., & Moscato, P. (2017, June). *PasMoQAP: a parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem. In 2017 IEEE congress on evolutionary computation (CEC)* (pp. 1103-1110). IEEE.

Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *Siam Review*, 3(1), 37-50.

Taillard, É. (1991). Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4-5), 443-455.

Tosun, U. (2015). On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. *Engineering applications of artificial intelligence*, 39, 267-278.

Tsutsui, S., & Fujimoto, N. (2009, July). Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers* (pp. 2523-2530).

Xia, Y., & Yuan, Y. X. (2006). A new linearization method for quadratic assignment problems. *Optimisation Methods and Software*, 21(5), 805-818.

Zhang, H., Beltran-Royo, C., & Ma, L. (2013). Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. *Annals of Operations Research*, 207(1), 261-278.

Zhao, Q., Karisch, S. E., Rendl, F., & Wolkowicz, H. (1998). Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1), 71-109.