Journal of Global Optimization **15:** 127–156, 1999.
© 1999 *Kluwer Academic Publishers. Printed in the Netherlands.*

Generating Hard Test Instances with Known Optimal Solution for the Rectilinear Quadratic Assignment Problem

G. PALUBECKIS
Associate Professor, Department of Practical Informatics, Kaunas University of Technology, Studentu 50, 3031 Kaunas, Lithuania
(e-mail: gintaras@soften.ktu.lt)

(Received 6 February 1998; accepted in revised form 27 October 1998)

Abstract. In this paper we consider the rectilinear version of the quadratic assignment problem (QAP). We define a class of edge-weighted graphs with nonnegatively valued bisections. For one important type of such graphs we provide a characterization of point sets on the plane for which the optimal value of the related QAP is zero. These graphs are used in the algorithms for generating rectilinear QAP instances with known provably optimal solutions. The basic algorithm of such type uses only triangles. Making a reduction from 3-dimensional matching, it is shown that the set of instances which can be generated by this algorithm is hard. The basic algorithm is extended to process graphs larger than triangles. We give implementation details of this extension and of four other variations of the basic algorithm. We compare these five and also two existing generators experimentally employing multi-start descent heuristic for the QAP as an examiner. The graphs with nonnegatively valued bisections can also be used in the construction of lower bounds on the optimal value for the rectilinear QAP.

Key words: Test instances, Quadratic assignment, Graphs, Combinatorial optimization

1. Introduction

Given a finite set $N = \{1, 2, \dots, n\}$ and three $n \times n$ matrices $W = (w_{ij})$, $D = (d_{ij})$ and $\Psi = (\Psi_{ij})$ with real entries, the *quadratic assignment problem* (QAP) is to find a permutation p of the set N such that the sum

$$f(p) = \sum_{i \in N} \sum_{j \in N} w_{ij} d_{p(i)p(j)} + \sum_{i \in N} \psi_{ip(i)} \tag{1}$$

is minimized. When the matrix D is composed of the shortest rectilinear distances between pairs of n points in the Euclidean space, the problem given by (1) is called *rectilinear QAP*. This problem is an important case of the *Euclidean QAP* in which the entries of the matrix D are required to fulfill the triangle inequality. We assume in this paper that the space from which the points are taken is two dimensional, i.e., the plane. We also assume without loss of generality that the lower half of W under the main diagonal is zero, i.e., $w_{ji} = 0$ for all i, j such that $i<j$. Indeed, if

$w_{ji} \neq 0$ for some pair i, j, $i<j$, then we can replace w_{ij} with $w_{ij}+w_{ji}$ (since D is a symmetric matrix). Also, the main diagonal of the matrix W can be made zero because such is the main diagonal of D.

A typical example of the Euclidean QAP is the facility location problem, in which n given facilities are to be assigned to the same number of locations. In this interpretation, D is the matrix of distances, maybe rectilinear, between locations, and $W = (w_{ij})$ is the flow matrix, i.e., w_{ij} is the flow of materials from facility i to facility j. The cost of simultaneously assigning facility i to location k and facility j to location l is given by the product $w_{ij}d_{kl}$. The fixed cost of assigning facility i to location k is given by the entry Ψ_{ik} of the matrix Ψ . The objective is to find a one-to-one assignment of n facilities to n locations, i.e., a permutation p, such that the total cost of the assignment is minimized. For other applications of the QAP, frequently its Euclidean version, see the reviews by Burkard (1984), Finke, Burkard and Rendl (1987), Pardalos, Rendl and Wolkowicz (1994), Burkard, Çela, Pardalos and Pitsoulis (1998), and the recent book by Çela(1998).

The QAP is an NP-hard problem for which exact algorithms are able to solve only instances of size less than 25 (see, e.g., Mautor and Roucairol, 1994; Clausen and Perregaard, 1997). So, for larger QAP instances different heuristics are widely used. The existing heuristic algorithms for the QAP include those based on simulated annealing (Burkard and Rendl, 1984; Wilhelm and Ward, 1987; Connolly, 1990), genetic (Fleurent and Ferland, 1994; Tate and Smith, 1995), tabu search (Skorin-Kapov, 1990; Taillard, 1991; Battiti and Tecchiolli, 1994), greedy randomized adaptive search (Li, Pardalos and Resende, 1994; Pardalos, Resende and Li, 1996; Pardalos, Resende and Pitsoulis, 1997; see also Feo and Resende, 1995) and ant system (Gambardella, Taillard and Dorigo, 1997) techniques. Usually, the heuristic algorithms are tested using randomly generated QAP instances or standard test problems in the literature. The latter are collected into a special library, called QAPLIB (Burkard, Karisch and Rendl, 1997). However, both for larger problems in QAPLIB and for larger instances produced by traditional generators the optimal solutions are not known. Testing and evaluation of a heuristic is more complete when, together with benchmarks from QAPLIB, instances with an apriori known optimal solution or just optimal value were used. In this context, the design of special generators producing such instances is an important problem. For the rectilinear QAP such a generator was proposed by Palubeckis (1988). Later Li and Pardalos (1992) provided a general schema and two generators based on it for arbitrary (not necessarily Euclidean) QAP. The instances created by these generators were used to test three heuristics for the QAP: simulated annealing (Burkard and Rendl, 1984), tabu search (Skorin-Kapov, 1990) and graph-partitioning (Murthy, Pardalos and Li, 1992). The computational results provided by Li and Pardalos (1992) show that such instances are not easy for these heuristics with regard to both the solution quality and solution time.

Given a generator for the QAP or any other problem in combinatorial optimization, the following question is very important: whether the set of instances produced by this generator is hard or not. The definition of hardness was suggested by Sanchis (1990). The set is hard if no polynomial-time algorithm solves each problem in this set, unless P = NP (a more formal definition of hardness is given in the next section). The generators with hard output sets were developed for several optimization problems on graphs, for example, for the well-known maximum clique problem (Sanchis and Jagota, 1996; see also Hasselberg, Pardalos and Vairaktarakis (1993) for an implementation of such a generator). In this paper we present several algorithms for generating instances with known optimal solution for the rectilinear QAP and compare them experimentally with existing generators. These algorithms are similar to the generator proposed by Palubeckis (1988). As shown by Cyganski, Vaz and Virball (1994), the decision problem for QAP instances produced by this generator, and even its generalizations, is polynomially solvable. On the other hand, the computational complexity of these instances in a sense of the definition of hardness given above is actually an open question and will be settled in the current paper – we shall prove that the problem of finding an optimal solution for these instances is NP-hard.

The paper is organized as follows. In Section 2 we introduce some preliminaries. Particularly, we define a class of edge-weighted graphs with nonnegatively valued bisections and show that the optimal value of the quadratic assignment problems related to these graphs is nonnegative. For one important type of such graphs we provide a characterization of point sets on the plane for which this bound is tight. In Section 3 we present our basic algorithm for generating rectilinear QAP instances with known optimal solutions. To construct a distance matrix, the algorithm uses some subset of points taken from the 2-dimensional grid. The flow matrix is obtained as a composition of matrices corresponding to triangles having one negative and two positive edges. The matrix Ψ in this and other generators considered in this paper is zero. We prove that the set of instances which can be generated by this algorithm is hard. For this we use 3-dimensional matching – the well-known NP-complete problem. We also prove that the set remains hard for the grid of size (asymptotically) close to the number of objects. We extend the basic generator to process graphs larger than triangles. The details of this extension together with three implementations of the basic algorithm and one additional generator are provided in Section 4. The results of experimental comparison of these and also two existing generators are given in Section 5. The results are obtained using multi-start descent heuristic for the QAP. In Section 6 we remark that the graphs with nonnegatively valued bisections can also be used in the construction of lower bounds on the optimal value for the rectilinear QAP.

2. Definitions and preliminaries

In this section we present some definitions and basic facts used in the construction and characterization of QAP instances.

We denote a graph by $G = (V, E)$ where V is the set of vertices and E is the set of edges (unordered pairs of vertices), and a digraph by $B = (V, A)$ where V is the set of vertices as before and A is the set of arcs (ordered pairs of vertices). All the graphs and digraphs considered in this paper are without loops. Usually, the edges of a graph $G = (V, E)$ will be supplied with weights c_{ij}, $(i, j) \in E$. We assume throughout the paper that c_{ij} and c_{ji} denote the same object – the weight of the edge $(i, j) \in E$. A *path* $P_k(v, v')$ of length $k \geqslant 1$ in $B = (V, A)$ is a digraph with vertex set $\{v_1 = v, v_2, \ldots, v_{k+1} = v'\}$, $v_i \in V$, $i = 1, \ldots, k+1$, and arc set $\{(v_i, v_{i+1}) | i = 1, \ldots, k\}$.

Given a flow matrix W, we can construct a graph $G(W) = (V(W), E(W))$ whose vertex set corresponds to the set $\{i | i \in N$ and there is $j \in N$ such that $w_{ij} \neq 0$ or $w_{ji} \neq 0\}$, whose edge set $E(W) = \{(i, j) | i, j \in N, w_{ij} \neq 0\}$, and whose edge weights $c_{ij} = w_{ij}$, $(i, j) \in E(W)$. Conversely, to any graph $G = (V, E)$, $V \subseteq N$, we can associate an $n \times n$ flow matrix $W(G)$ with nonzero entries defined by the edges of G, i.e., with $w_{ij} = c_{ij}$ (if $i < j$) or $w_{ji} = c_{ij}$ (if $i > j$) if $(i, j) \in E$, and $w_{ij} = 0$ otherwise.

All the distance matrices considered in this paper are defined by some set of points on the plane. We will write $D(S)$ to denote such a matrix for a set S. For $s = (x, y)$, $s' = (x', y') \in S$ the corresponding entry of $D(S)$ is equal to $d(s, s') = |x - x'| + |y - y'|$. The sum of all entries of D is denoted by $\Sigma(D)$.

Given integers $n_x \geqslant 1$, $n_y \geqslant 1$, we define a *regular 2-dimensional* (or, more precisely, regular $n_x \times n_y$) *grid* $Q(n_x, n_y)$ as a set $\{(i, j) | i = 1, \ldots, n_x, j = 1, \ldots, n_y\}$ of points on the plane. The *size* of the grid $Q(n_x, n_y)$ is the product $n_x n_y$.

We denote the optimal value of (1) with zero Ψ by $f_0(W, D)$.

We use the following definition taken from Sanchis (1990).

DEFINITION 1. Let Π be an NP-hard optimization problem. A set I of instances of Π is *hard* with respect to Π if no polynomial-time approximation algorithm for Π can give the optimal answer for all instances in I, unless P = NP.

The following obvious fact describes a decomposition principle the reverse of which, i.e. composition, stands at the basis of the instance construction procedure.

LEMMA 1. *If $W = W_1 + W_2$ and D is a distance matrix, then*

$$f_0(W, D) \geqslant f_0(W_1, D) + f_0(W_2, D).$$

Note that this lemma holds for an arbitrary distance matrix, not necessarily the rectilinear one.

Clearly, if some solution p is optimal for each of the problems corresponding to W_1 and W_2, then p is also optimal for the initial problem defined by the matrix W.

Next we introduce a special class of edge-weighted graphs and give a bound on the optimal value of f for members of this class.

DEFINITION 2. A graph $G = (V, E)$ with edge weights c_{ij}, $(i, j) \in E$, is a *PB-graph* (has nonnegatively valued bisections) if the sum $\Omega(G, V')$ of the weights in the set $\{c_{ij} | (i, j) \in E, i \in V', j \in V \backslash V'$ or $i \in V \backslash V', j \in V'\}$ is nonnegative for each subset $V' \subset V$.

LEMMA 2. *If W is such that $G(W)$ is a PB-graph, then for any rectilinear distance matrix D*

$$f_0(W, D) \geqslant 0. \tag{2}$$

Proof. Suppose we are given n points defining the matrix D, and let $x_1, \dots, x_n$ be their x-coordinates sorted nondecreasingly. This sorting defines some ordering of the points. Let S_{ix}, $i \in \{1, \dots, n-1\}$, denote the set consisting of the first i points in this ordering. Let $y_1, \dots, y_n$ and S_{jy}, $j = 1, \dots, n-1$, be the list and sets defined analogously with respect to the y-axis. Using the expression for the entry of the rectilinear distance matrix we can write

$$f_0(W, D) = \sum_{i=1}^{n-1} \Omega(G(W), V_{ix})(x_{i+1} - x_i) + \sum_{j=1}^{n-1} \Omega(G(W), V_{jy})(y_{j+1} - y_j)$$

where V_{ix} (respectively, V_{jy}) is the set of the vertices of $G(W)$ corresponding to objects assigned by an optimal permutation to the points in S_{ix} (respectively, S_{jy}). The nonnegativity of the optimal value now follows from the inequalities $x_{i+1} \geqslant x_i$, $y_{j+1} \geqslant y_j$, $i, j = 1, \dots, n-1$, and $\Omega(G(W), V') \geqslant 0$ for any vertex subset V' and thus for V_{ix} and V_{jy}, $i, j = 1, \dots, n-1$. □

In this paper we are interested in PB-graphs which are signed graphs, that is, have all edge weights equal to 1 or -1. Perhaps, the simplest signed PB-graphs are cycles with one negative edge and the remaining edges being positive. The smallest such cycle, namely, the triangle with one negative and two positive edges is used in generators described in Palubeckis (1988) and Li and Pardalos (1992). Another type of signed PB-graphs can be defined as follows. Let l, $l \geqslant 3$, be the number of vertices of a graph. The vertex set V is divided into two subsets V_1, V_2 such that $|V_1| = \lceil l/2 \rceil$, $|V_2| = l - |V_1|$. The edge set E is complete and consists of the subset $E_+ = \{(i, j) | i \in V_1, j \in V_2\}$ of positive edges and the subset $E_- = \{(i, j) | i, j \in V_1$ or $i, j \in V_2\}$ of negative edges. We denote this graph by $H_l = (V, E)$. It is easy to verify that H_l for any $l \geqslant 3$ satisfies the condition given by Definition 2, so is a PB-graph.

For $l \geqslant 3$, let $W^l = W(H_l)$ denote the flow matrix corresponding to H_l. In the construction of QAP instances we use H_l, $l \geqslant 3$, together with a point set S for which $f_0(W^l, D(S)) = 0$. Now we address the problem of characterization of point sets having this property.

Let $S = \{s_i = (x_i, y_i) | i = 1, \dots, l\}$ be a set of points on the plane, and $X = \{X_j | j = 1, \dots, q\}$ (respectively, $Y = \{Y_k | k = 1, \dots, r\}$) be the minimal

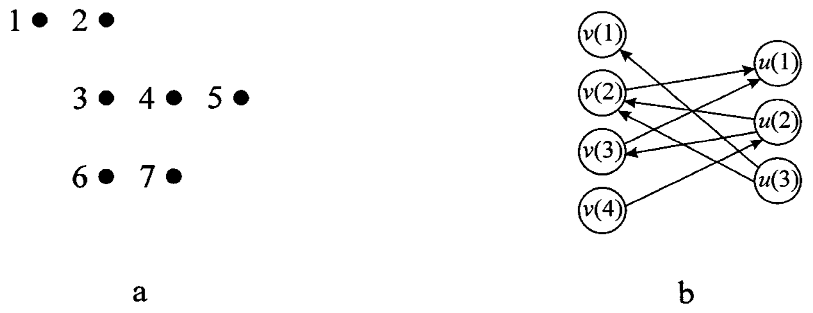

Figure 1. Directed graph $B(S, p)$ (on the right) for set S and identity permutation p (on the left).

increasingly ordered set of vertical (respectively, horizontal) lines on which points of S are located. So, each point is defined by some pair of lines, one vertical and another horizontal, and each line contains at least one point of S. For a point $s_i \in S$, we denote this pair by $X(s_i) \in X$, $Y(s_i) \in Y$. Given a point set S and some assignment p of the vertices of $H_l = (V_1 \cup V_2, E)$ to the points in S, we can construct a directed graph $B(S, p) = (V = V(X) \cup U(Y), A)$ whose vertex subsets $V(X)$ and $U(Y)$ correspond to X and Y, respectively, i.e., $V(X) = \{v(X_j) | j = 1, \dots, q\}$, $U(Y) = \{u(Y_k) | k = 1, \dots, r\}$, where $v(X_j)$, $j \in \{1, \dots, q\}$, (respectively, $u(Y_k)$, $k \in \{1, \dots, r\}$) is the vertex associated with the vertical line $X_j \in X$ (respectively, with the horizontal line $Y_k \in Y$), and whose arc set A corresponds to p and is the union of the following disjoint sets: $\{(v(X_j), u(Y_k)) | X_j = X(s_i), Y_k = Y(s_i)$ for some $s_i \in S$, and $p(v) = i$ for some $v \in V_2\}$, $\{(u(Y_k), v(X_j)) | X_j = X(s_i)$, $Y_k = Y(s_i)$ for some $s_i \in S$, and $p(v) = i$ for some $v \in V_1\}$. Thus for two different assignments p and p' the digraphs $B(S, p)$ and $B(S, p')$ may differ only in orientation of some of their arcs. We denote by $\delta_{\text{in}}(X_j)$ (respectively, $\delta_{\text{out}}(X_j)$) the number of arcs in A whose head (respectively, tail) is the vertex $v(X_j)$. So, $\delta(X_j) := \delta_{\text{in}}(X_j) + \delta_{\text{out}}(X_j)$ is the total number of arcs incident to $v(X_j)$. The values $\delta(Y_j)$, $\delta_{\text{in}}(Y_j)$ and $\delta_{\text{out}}(Y_j)$ are defined analogously.

EXAMPLE 1. Consider the set $S = \{(1, 3), (2, 3), (2, 2), (3, 2), (4, 2), (2, 1), (3, 1)\}$. For this set, $X = \{1, 2, 3, 4\}$, $Y = \{1, 2, 3\}$. For illustration of the digraph we can choose the identity permutation $p = (1, 2, \dots, 7)$ defined on the set $V = V_1 \cup V_2$, $V_1 = \{1, 2, 3, 4\}$, $V_2 = \{5, 6, 7\}$. Using p, vertex $i \in V$ is assigned to the point $s_{p(i)} = s_i \in S$ (see Figure 1a). The digraph $B(S, p)$ for the set S and permutation p is shown in Figure 1b, where the vertices in $V(X)$ (respectively, in $U(Y)$) form the left (respectively, right) column. It should be clear that this digraph represents a group of permutations obtained from p by permuting the first 4 and the last 3 elements of p independently.

Now we will characterize the digraphs B for which the underlying assignment p is optimal and the optimal value is 0. Let $J(X)$ denote the set of indices $j \in \{1, \dots, q\}$ such that $\delta(X_j)$ is odd. Obviously, $|J(X)|$ is even if and only if l is even.

Assuming $J(X) = \{j_1, \dots, j_u\}$, $u \geqslant 1$, for odd l, define $J_1(X) = \{j_1, j_3, \dots, j_u\}$ (or $J_1(X) = \{j_1\}$ if $u = 1$) and $J_2(X) = \{j_2, j_4, \dots, j_{u-1}\}$ (or $J_2(X) = \emptyset$ if $u = 1$). Let $J(Y)$, $J_1(Y)$, $J_2(Y)$ be the analogous sets with respect to the y-coordinate. Define $\Delta(X_j) = \delta_{\text{in}}(X_j) - \delta_{\text{out}}(X_j)$, $\Delta(Y_k) = \delta_{\text{in}}(Y_k) - \delta_{\text{out}}(Y_k)$. $\Delta(X_j) = 0$ means that a half of the vertices in the set $\{v \in V | s_{p(v)}$ is on $X_j\}$ belongs to V_1 and the other half to V_2. Furthermore, for odd l the sum of $\Delta(X_j)$ taken over the index set $\{1, \dots, q\}$ is equal to 1.

LEMMA 3. *Let S be a set of l points on the plane. A permutation p described by the digraph $B(S, p)$ is optimal and $f_0(W^l, D(S)) = 0$ if and only if:*

$$\begin{aligned} &\textit{for } l \textit{ even}, \quad \Delta(X_j) = 0 \quad \textit{for each } j = 1, \dots, q, \textit{ and} \\ &\qquad\qquad\quad\ \Delta(Y_k) = 0 \quad \textit{for each } k = 1, \dots, r; \end{aligned}$$

$$\textit{for } l \textit{ odd}, \quad \Delta(X_j) = \begin{cases} 1 & \textit{if } j \in J_1(X) \\ -1 & \textit{if } j \in J_2(X) \\ 0 & \textit{if } j \in \{1, \dots, q\} \backslash J(X) \end{cases} \tag{3}$$

and

$$\Delta(Y_k) = \begin{cases} -1 & \textit{if } k \in J_1(Y) \\ 1 & \textit{if } k \in J_2(Y) \\ 0 & \textit{if } k \in \{1, \dots, r\} \backslash J(Y). \end{cases} \tag{4}$$

Proof. Consider a vertical line specified by the x-coordinate x greater than that for X_1 and smaller than that for X_q. We can assume that this line is different from the lines in the set X. Let l_1 (respectively, l_2) be the number of vertices in V_1 (respectively, in V_2) which are assigned by p to the points of S at the left of the line x. Conditions of the lemma imply that either $l_1 = l_2$ or $|V_1| - l_1 = |V_2| - l_2$. So the number of positive edges crossing the line x is equal to the number of negative edges crossing the same line. The same holds for y-direction as well. Therefore, $f(p) = 0$, and by (2) the solution p is optimal.

To prove the 'only if' part, suppose that p is an optimal permutation with zero optimal value, but the conditions of the lemma for the corresponding digraph $B(S, p)$ are violated. We will derive a contradiction to this statement. Without loss of generality, we may assume that the value of Δ is different than that required by the lemma for at least one vertical line in X. Let q' be the number of such lines and X_i be the leftmost of them. Since by the definition of the set A and function Δ the sum $\sum_{j=1}^{q} \Delta(X_j) = |V_1| - |V_2| = 0$ or 1 depending on the parity of l, it follows that $q' \geqslant 2$. Therefore, $i < q$.

Consider any vertical line, say x', located strictly between X_i and X_{i+1}. Let l_1, l_2 have the same meaning as before, except that now these numbers are defined with respect to the line x' instead of x. Clearly, if l is even, then $l_1 \neq l_2$. If l is odd, then either $l_1 \geqslant l_2 + 2$ if $\Delta(X_i)$ is greater than the corresponding constant on the right-hand side of (3), or $l_1 < l_2$ otherwise. For assignment p, the total weight of edges of H_l crossing the line x' is equal to $(l_1 - l_2)(l_1 - l_2 + \lfloor l/2 \rfloor - \lceil l/2 \rceil)$. For l_1, l_2 specified above this expression is positive implying positivity of $f(p)$, a contradiction to our initial assumption. $\square$

Now assume that l is odd. We call a vertex $v \in B(S, p)$, $v = v(X_j)$ or $u(Y_k)$, *neutral* if the corresponding difference $\Delta = \Delta(X_j)$ or $\Delta(Y_k)$ is given by (3) or by (4); otherwise we call v *positive* or *negative* depending on whether Δ is greater or smaller than the corresponding value on the right-hand side of (3) or (4). We provide an algorithm for searching an assignment p satisfying the conditions of Lemma 3. In fact, the algorithm requires the condition (4) to be met for the initial assignment p already. To get such an assignment, we run through the set Y, taking the line Y_1 first, then Y_2 next and so on. Let Y_k stand for the line selected and m_k be the number of points on it. If $k \in J_1(Y)$, then we set $m'_k := \lceil m_k/2 \rceil$, otherwise $m'_k := \lfloor m_k/2 \rfloor$. We assign any m'_k vertices of V_1, which are free (not assigned earlier), to arbitrary m'_k points on Y_k and any $m_k - m'_k$ free vertices of V_2 to the rest points on Y_k. Upon termination of this procedure we have an assignment, called *y-compatible*, for which (4) is satisfied. So, the algorithm given below tries to balance Δ for the vertices in the set $V(X)$ only. The input besides digraph $B(S, p)$ includes also the set S. The algorithm goes as follows.

Algorithm BALANCE

1. Set $B_{\text{current}} := B(S, p)$.
2. For $B_{\text{current}} = (V(X) \cup U(Y), A)$, check whether the set $V_-(X) = \{v(X_j) \in V(X) \mid v(X_j)$ is negative$\}$ is nonempty. If so, take any $v = v(X_j) \in V_-(X)$ and go to 3. Otherwise go to 5.
3. Try to find a path in B_{current} between v and some positive vertex. If failure, then go to 5. Otherwise letting $P(v, v')$ be the path found proceed to 4.
4. Change an orientation of each arc on $P(v, v')$ to the opposite and return to 2.
5. Assign arbitrarily the vertices of H_l in the subset V_1 to points in $\{s_i \in S \mid (u(Y_k), v(X_j)) \in A,\ X_j = X(s_i),\ Y_k = Y(s_i)\}$ and the vertices in V_2 to the remaining points in S. Stop with this assignment.

We say that the algorithm ends with a positive answer if it stops when B_{current} has no negative vertex. We now estimate the computational complexity of the algorithm. First note that $B(S, p)$ has l arcs, so each step has at most linear running time. Let ξ_j be the constant on the right-hand side of (3). Each execution of the loop Step 2-Step 4 diminishes the sum $\sum_{j=1}^{q} |\Delta(X_j) - \xi_j|$ by 4 (since for the first vertex of the path the value of Δ is increased and for the last is decreased by 2). But this sum can be bounded from above by $q + \sum_{j=1}^{q} \delta(X_j) \leqslant 2l$. Thus, the algorithm runs in time $O(l^2)$.

The following pair of conditions provides a characterization of point sets for which the equality in (2) with respect to W^l is obtained.

THEOREM 1. *Let S be a set of $l \geqslant 3$ points on the plane and X (respectively, Y) be the set of vertical (respectively, horizontal) lines passing through the points in S. $f_0(W^l, D(S)) = 0$ if and only if:*

for l even, each line in $X \cup Y$ contains an even number of points of S;

for l odd, the algorithm BALANCE applied to any y-compatible assignment ends with a positive answer.

Proof. First consider the case of odd l. Suppose on contrary that for some assignment p_0 $f(p_0) = 0 = f_0(W^l, D(S))$ yet the algorithm stops with some assignment p' having at least one negative vertex in the corresponding digraph $B(S, p') = B_{\text{current}}$. From the definition of the digraph and classification of its vertices into neutral, positive and negative it follows that $B(S, p')$ must also contain at least one vertex which is positive.

For $B(S, p') = (V, A)$, define $A^* = \{a \in A|$ orientation of a in $B(S, p')$ is different from that in $B(S, p_0)\}$. Let B^* be the subdigraph of $B(S, p')$ induced by the arc set A^*. Suppose that Δ is defined with respect to B^*. Clearly, for a vertex $v = v(X_j)$ the following holds: $\Delta(X_j) < 0$ if v is negative (in $B(S, p')$), $\Delta(X_j) > 0$ if v is positive, and $\Delta(X_j) = 0$ if v is neutral. In addition, $\Delta(Y_k) = 0$ for any vertex $u(Y_k)$, $k \in \{1, \dots, r\}$.

Let v_1 be any negative vertex of B^*. Take some arc $(v_1, v_2) \in A^*$ and consider the vertex v_2. This vertex is neutral and, therefore, necessarily has at least one outcoming arc. Append any such arc, say (v_2, v_3), to (v_1, v_2). Continuing this way (under the restriction that each arc can be taken only once) we obtain some path $P_t(v_1, v_t)$ ending at some positive vertex. Note that the last vertex in this path cannot be neutral or negative since such vertices have at least as many outcoming arcs as incoming. Clearly, the path P_t is a subgraph of $B(S, p')$ and can be found in Step 3 of BALANCE, a contradiction to our initial assumption.

The ‘if’ part of the proof for odd l follows directly from Lemma 3.

Suppose now that l is even and the condition of the theorem is satisfied. We can consider a solution p for which $\Delta(Y_k) = 0$ for each $k = 1, \dots, r$. Similarly as in the case of odd l we call a vertex $v(X_j)$ negative if $\Delta(X_j) < 0$ and positive if $\Delta(X_j) > 0$. If p is such that $f(p) > 0$, then the digraph $B(S, p)$ always contains a path from any negative vertex to some positive one. Reversing an orientation of each arc on this path we obtain a new assignment with the sum of negative Δ closer to 0. Repeating this operation would result in the assignment p with $f(p) = 0$. If the condition of the theorem does not hold, then for any assignment the difference $\Delta(X_j) \neq 0$ for at least one index in $\{1, \dots, q\}$, and thus by Lemma 3 $f_0 > 0$. The proof is complete. □

COROLLARY. *For H_3 and S consisting of three points the optimal value $f_0 = 0$ if and only if one of the points in S lies inside the minimal rectangle enclosing the other two points of S.*

Proof. We will characterize digraphs $B(S, p)$ describing y-compatible assignments for which BALANCE fails to end with a positive answer. If $| X |= 2$ or

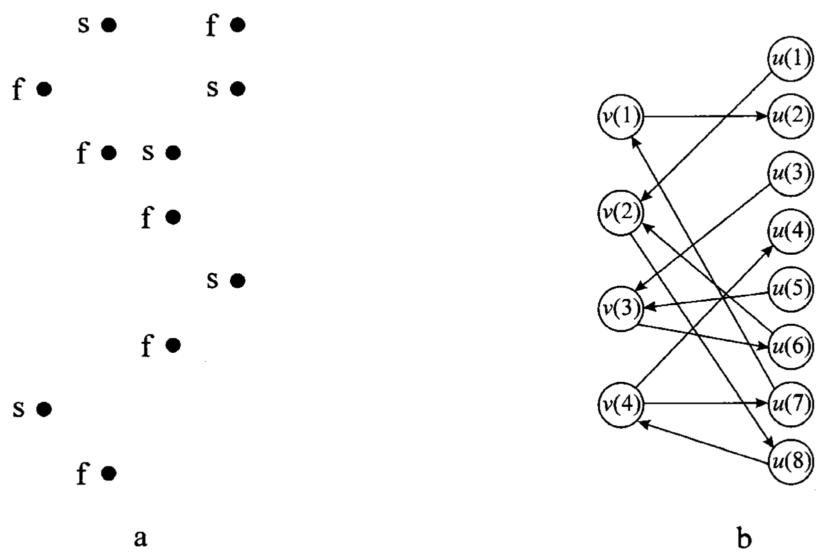

Figure 2. Set of points (a) and directed graph (b) for which BALANCE fails.

3, | Y |= 3, then such digraphs must have an arc $(v(X_j), u(Y_2))$, $j \in J_1(X)$. If | X |= 3, | Y |= 2, then they must have an arc $(u(Y_k), v(X_2))$, $k \in J_1(Y)$. Using definition of the set J_1 for X and Y we can ascertain that in each of these cases the condition stated in the corollary is violated. In all other cases BALANCE ends with a positive answer, and as can be readily checked the condition of the corollary is satisfied. □

REMARK 1. Checking conditions of the theorem is polynomial in time. For odd l, this follows from the polynomiality of the algorithm BALANCE.

EXAMPLE 2. Consider the following set of 11 points: (1,2), (1,7), (2,1), (2,6), (2,8), (3,3), (3,5), (3,6), (4,4), (4,7), (4,8). For this set, $X = \{1, 2, 3, 4\}, Y = \{1, 2, \ldots, 8\}$. Assign the vertices of H_{11} in the subset V_1 to points (1,7), (2,1), (2,6), (3,3), (3,5), (4,8) (in Figure 2a these points are marked “f”) and the vertices in V_2 to the remaining points (in Figure 2a marked “s”). It is easy to see that this assignment is y-compatible. The corresponding digraph is shown in Figure 2b. Clearly, $J_1(X) = \{2, 4\}$, $J_2(X) = \{3\}$. Checking (3) we can conclude that the vertex $v(4)$ is negative and the vertex $v(3)$ is positive. However, there are no path between these vertices (from $v(4)$ to $v(3)$), so for this set $f_0 > 0$.

3. Hard sets of QAP instances

In this section we first give a description of our main set of instances with known optimal solutions and prove that this set is hard (in a sense of Definition 1 with QAP in the role of Π). Then we evaluate asymptotically the minimal size of the grid used in the construction of these instances. Finally we formulate the set of less simple instances and show that this set is hard too.

Let n denote the number of points as before and let $Q(n_x, n_y)$ be 2-dimensional grid of size $n_x n_y \geqslant n$. Define $\theta = \{(i, j; k) | i, j, k \in N$ are pairwise different$\}$. Separating the last element in the triplet $(i, j; k) \in \theta$ from the first two means that $(i, j; k)$ and $(j, i; k)$ denote the same member of θ. To each triplet $(i, j; k) \in \theta$ and any nonzero scalar α we can associate the graph-triangle $T(i, j, k, \alpha)$ induced by the edges (i, k), (j, k) of weight α and the edge (i, j) of weight $-\alpha$. In fact, $T(i, j, k, \alpha)$ is a weighted version of H_3 with $V = \{i, j, k\}$. Given $Q(n_x, n_y)$, n, $n \leqslant n_x n_y$, and positive integers t and w, we can construct an instance of QAP using the following procedure.

Algorithm GENERATE

1. Choose n points $s_1, \ldots, s_n$ on the grid $Q(n_x, n_y)$. Set W equal to the all-zeros matrix. Let $\theta' = \{(i, j; k) \in \theta | d(s_i, s_k) + d(s_j, s_k) = d(s_i, s_j)\}$.
2. Repeat t times the following operations: choose a triplet $(i, j; k) \in \theta'$ such that $w_{ik}, w_{ki}, w_{jk}, w_{kj}$ are nonnegative and w_{ij}, w_{ji} nonpositive, allowing possible repetition with some triplet (or triplets) chosen earlier, and an integer $\alpha \in [1, w]$; set $W := W + W(T(i, j, k, \alpha))$.
3. Sort the points s_i, $i \in N$, in the order of appearance while scanning the rows of the grid $Q(n_x, n_y)$ sequentially. Construct the distance matrix $D(\{s_1, \ldots, s_n\})$ in such a way that for any $i \in N$ the ith row and column of D would correspond to the ith point in the order obtained.
4. Add $w_{\min} := -\min\{w_{ij} | i, j \in N, i < j\}$ to each entry of W above the main diagonal. Stop with the matrices W and D.

This algorithm is quite similar to the algorithm proposed by Palubeckis (1988). Both construct the flow matrix in the same way – by choosing triplets defining the triangles with one negative and two positive edges and modifying the entries of W corresponding to these edges. To make the matrix W nonnegative, some constant to its entries is added. In both cases the matrix D is defined by some set of grid points, and the optimal value is computed according to the same formula. The main difference between these algorithms lies in the triplet selection strategy. In the old algorithm triplets $(i, j; k) \in \theta'$ with the largest distance $d(s_i, s_j)$ are considered first. When some triplet, say (i', j', k'), is selected, then all triplets in the set $\{(i', j'; k) \in \theta' | k \neq k'\}$ become inadmissible. In the algorithm GENERATE these restrictions are removed, and any triplet in θ' satisfying the condition stated in Step 2 (this condition is satisfied in the old algorithm as well) can be chosen.

PROPOSITION 1. *For $i \in N$ let l_i be the rank of the point s_i in the sequence obtained in Step 3 of GENERATE. For a QAP defined by the pair (W, D) delivered by GENERATE, the assignment $p = (l_1, \ldots, l_n)$ is optimal, and the optimal value is equal to $w_{\min}\Sigma(D)/2$.*

Proof. Consider the flow matrix $W(T(i, j, k, \alpha))$ corresponding to a triplet $(i, j; k)$ selected in Step 2 of GENERATE. It is clear from the definition of the set θ' and Corollary to Theorem 1 that for this matrix and distance matrix D the assignment p is optimal with the value $f(p) = 0$. The claim is established by applying Lemma 1 recursively for the set of t triplets and observing that Step 4 of the algorithm does not violate the optimality of p. □

In an implementation of the algorithm GENERATE some mechanisms for making choice of the set of points in Step 1 and triplets in Step 2 should be specified. We postpone this question until the next section.

Let $I(n, n_x, n_y, t, w)$ denote the set of pairs (W, D) which can be obtained using algorithm GENERATE. We will show that for a wide range of values of n_x, n_y, t and w this set is hard. We use a restricted version of the following NP-complete problem (Garey and Johnson, 1979).

3-Dimensional Matching (3DM)

Instance. Disjoint sets U_1, U_2, U_3 with $|U_1| = |U_2| = |U_3| = m$ and a set of triplets $Z \subseteq U_1 \times U_2 \times U_3$.

Question. Does there exist an exact matching $M \subseteq Z$, i.e., $|M| = m$ and each element of $U = U_1 \cup U_2 \cup U_3$ occurs in exactly one triplet of M?

This problem remains NP-complete even in the case when each element of U is included in at most three triplets of Z (see Garey and Johnson, 1979; also Dyer and Frieze, 1985) for a further restriction of 3DM). We use a slight embellishment of this version of 3DM.

Let $\beta(u_1)$ (respectively, $\beta(u_2)$ and $\beta(u_3)$) be the number of appearances of $u_1 \in U_1$ (respectively, $u_2 \in U_2$ and $u_3 \in U_3$) in the triplets of the set Z.

PROPOSITION 2. *3DM with $\beta(U_1)$=$\beta(U_2)$=$\beta(U_3)$=3 for each $u_1 \in U_1, u_2 \in U_2, u_3 \in U_3$ is NP-complete.*

Proof. We use a reduction from 3DM with $\beta(U) \in \{2, 3\}$ for each $u \in U$. The case when for some $u \in U$ $\beta(U) = 1$ can be excluded from consideration since the triplet containing such u always belongs to an exact matching (if any) and can be removed from Z (together with its elements from U). Suppose $\beta(u_1)$=2 for some $u_1 \in U_1$. Then necessarily exist $u_2 \in U_2$ and $u_3 \in U_3$ such that $\beta(u_2) = \beta(u_3)$=2. Assuming that $k_i \notin U$, $i = 1, 2, 3$, add k_1 to U_1, k_2 to U_2, k_3 to U_3, and (u_1, k_2, k_3), (k_1, u_2, k_3), (k_1, k_2, u_3), (k_1, k_2, k_3) to Z. It is easy to see that for the new problem the answer is 'yes' if and only if the answer 'yes' is for the initial one. This follows from the observation that the triplet (k_1, k_2, k_3) must belong to any exact matching for the new problem. Note also that now $\beta(u_1)$=$\beta(u_2)$=$\beta(u_3)$=3. Continuing this way we obtain the problem with each element included in exactly three triplets. With respect to the existence of an exact matching this problem is equivalent to the initial one, and thus is NP-complete. □

Now we state the main result of this section. Let $C_k = C_k(H_4^1, \ldots, H_4^k)$, $k \geqslant 2$, denote the chain of graphs $H_4^i = (V^i, E^i)$, $i = 1, \ldots, k$, each isomorphic to H_4, connected in such a way that $|V^i \cap V^{i+1}| = 2$ for each $i = 1, \ldots, k-1$, $V^i \cap$

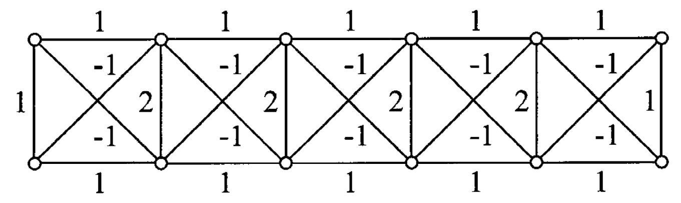

Figure 3. Chain $C_5 = C_5(H_4^1, \ldots, H_4^5)$.

$V^j = \emptyset$ for each pair $i, j = 1, \ldots, k, |i - j| > 1$, and the weight of the edge $e_i \in E^i \cap E^{i+1}$ for each $i = 1, \ldots, k - 1$, is 2, i.e., is obtained by summing the weights of this edge in H_4^i and H_4^{i+1} (it is required for this edge to be positive in both graphs). An example of the chain is shown in Figure 3.

In the formulation below we do not strive to give tight lower bounds on the dimensions of the grid n_x, n_y. Essentially, we only assert that QAP instances produced by GENERATE applied to a grid, whose dimensions are sufficiently large, are NP-hard. In fact, the reduction used in the proof of the theorem given below allows to decrease n_y at the cost of increasing n_x and vice versa. So, instead of trying to characterize the smallest n_x and n_y for which the reduction still holds it is more reasonable to evaluate the minimal size of the grid, that is, the product $n_x n_y$. Later in this section we will give an asymptotic estimate of such kind.

THEOREM 2. *For any $n_x \geqslant n/5$, $n_y \geqslant n/9$, $t \geqslant 2n$ and positive integer w all bounded from above by some polynomial of n the set $I(n, n_x, n_y, t, w)$ is hard.*

Proof. We will use a reduction from 3DM with each element of U included in exactly three triplets. So, in our case $|Z| = 3m$ (and certainly $m \geqslant 2$). We also suppose that the set Z does not have duplicate triplets. To make a construction of a QAP more lucid, we assume in the beginning that $b = n/(12m)$ is an integer. We first select the set S of points on the grid. The rows of the grid correspond to the elements of U as described by the following mapping: $\rho(u_{1i}) = \{2i - 1, 2i\}$, $\rho(u_{2i}) = \{2m+2i-1, 2m+2i\}$, $\rho(U_{3i}) = \{4m+2i-1, 4m+2i\}$, $u_{1i} \in U_1, u_{2i} \in U_2$, $u_{3i} \in U_3, i = 1, \ldots, m$. The first $6m$ columns correspond to the triplets in $Z : \mu(z_i) = \{2i - 1, 2i\}, z_i \in Z, i = 1, \ldots, 3m$. For $z_l = (u_{1i}, u_{2j}, u_{3k}) \in Z$ define $S(z_l) = \mu(z_l) \times (\rho(u_{1i}) \cup \rho(u_{2j}) \cup \rho(u_{3k}))$. The set S can be divided into two disjoint subsets S_1 and S_2. The first subset $S_1 = U_{l=1}^{3m} S(z_l)$. Figure 4 shows the set S for $m = 3$ and $Z = \{(u_{11}, u_{22}, u_{33}), (u_{11}, u_{21}, u_{31}), (u_{11}, u_{22}, u_{32}), (u_{12}, u_{21}, u_{33}), (u_{12}, u_{23}, u_{32}), (u_{12}, u_{23}, u_{31}), (u_{13}, u_{22}, u_{31}), (u_{13}, u_{21}, u_{32}), (u_{13}, u_{23}, u_{33})\}$. The subset S_1 consists of points in the columns 1 through 18. To define the second subset, we distinguish between the following two cases.

Case 1. $m \leqslant (b - 2)/3$. We append the grid obtained so far with the set $X = \{6m + 1, \ldots, 6m + 2(b - 3)\}$ of additional $2(b - 3)$ columns. In this case, $S_2 = \{1, \ldots, 6m\} \times X$.

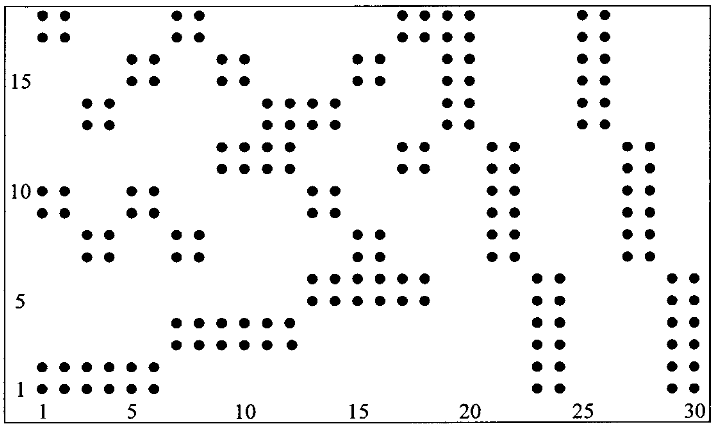

Figure 4. An example of the point set S used in the reduction.

Case 2. $m > (b-2)/3$. It is assumed that n is such that $b \geqslant 5$, i.e., $n \geqslant 60m$. We append the grid with $2q = 2\lceil 3m(b-3)/(b-2)\rceil$ new columns divided into pairs $\pi_1, \ldots, \pi_q$ using the mapping $\tau(i) = \pi_i = \{6m+2i-1, 6m+2i\}$, $i = 1, \ldots, q$. To each $u \in U$ we assign an $(b-3)$-element subset $\Pi(u)$ of $\{\pi_1, \ldots, \pi_q\}$. We require this assignment to be such that for any $i \in \{1, \ldots, q\}$ the cardinality of the set $\lambda_i = \{u \in U | \pi_i \in \Pi(u)\}$ is at most $b-2$. Let $\eta(u)$, $u \in U$, be the union of columns in all pairs $\pi_i \in \Pi(u)$. Define $S_2(u) = \rho(u) \times \eta(u)$. Then the subset $S_2 = \cup_{u \in U} S_2(u)$. In Figure 4 the subset S_2 consists of points in the columns 19 through 30.

In both cases it is easy to verify that $|S| = |S_1| + |S_2| = n$. To finish the construction, we use n-vertex graph G consisting of m copies of the chain C_5 and $3m$ copies of the chain C_{2B-3}. Each pair of chains in G is vertex disjoint. The weights of the edges of G belong to the set $\{-1, 1, 2\}$. We take the matrix $W = W(G)$ as a flow matrix of an instance of QAP in our reduction.

We argue that a given instance of 3DM has an exact matching if and only if $f_0(W(G), D(S))$=0. Suppose that for some assignment p $f(p) = 0$. It follows from the construction of G, definition of C_k and Lemma 2 that $f_0(W(G), D(S)) \geqslant 0$ and, therefore, such an assignment is optimal. Since G is a collection of chains we can write

$$f(p) = \sum_{i=1}^{m} f(5, i) + \sum_{i=1}^{3m} f(2b-3, i), \tag{5}$$

where $f(5, i)$ (respectively, $f(2b-3, i)$) is a term of $f(p)$ defined by the weights of the ith copy of C_5 (respectively, C_{2b-3}). Clearly, we must have $f(5, i) = 0$ for each $i = 1, \ldots, m$ and $f(2b-3, i) = 0$ for each $i = 1, \ldots, 3m$. We can apply Theorem 1 to each graph H_4^i, $i \in \{1, \ldots, k\}$, in $C_k = C_k(H_4^1, \ldots, H_4^k)$, $k = 5$ or $2b-3$. Since H_4^i and H_4^{i+1} for each $i = 1, \ldots, k-1$ have two vertices in common, we can see that $f(k, i) = 0$, $k = 5$ or $2b-3$, if and only if the set of points $\{s_{p(v)} | v$

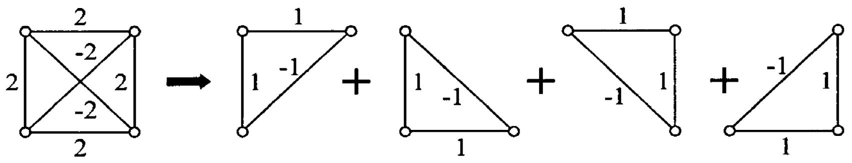

Figure 5. Decomposition of H_4 with double weights.

is a vertex of the ith copy of $C_k\}$ is defined by the intersection either of 2 grid columns and k+1 grid rows or 2 grid rows and k+1 grid columns. The latter is the only possible case for the copies of C_{2b-3}. This claim is established using the following facts. If $m \leqslant (b-2)/3$, the number of grid rows $6m < 2b-2$ and is too small to embed C_{2b-3} vertically. If $m > (b-2)/3$, then $2|\lambda_i| \leqslant 2(b-2) < 2b-2$ and, as assumed, $b \geqslant 5$, so the number of points on the vertical lines again is insufficient. After embedding $3m$ copies of C_{2b-3}, exactly two points of S in each grid row remain to be free. This means that the vertices of a copy of C_5 must be assigned by p to the points in the set $S(z_i)$ for some $z_i \in Z$. The collection of such sets used for m copies of C_5 spans all the rows of the grid. Consequently, the set of triplets $M = \{z_i \in Z | S(z_i)$ is used to embed some copy of $C_5\}$ is an exact matching for a given instance of 3DM.

The proof of the claim in the direction from the existence of exact matching to zero value of f_0 is straightforward.

For the moment, we will digress from the main stream of the proof in order to outline the changes in the reduction for the case when n is an arbitrary sufficiently large positive integer. If n is not divisible by $12m$, we use $b = \lfloor b'/4 \rfloor, b' = \lfloor n/(3m) \rfloor$, instead of b defined earlier. In this case we also define the remainder $r = n - 3mb' + 3m\sigma$, where $\sigma \in \{0, 1\}$, and $\sigma = 1$ if and only if $b' \equiv 1$ or 3(mod 4). Clearly $r < 6m$.

If $b' \equiv 0$ or 1(mod 4), we simply append the grid with additional column containing r points of S and add r isolated vertices to the graph G constructed in the same way as before.

So, let $b' \equiv 2$ or 3(mod 4). To specify the analogues of Cases 1 and 2 considered above, we can use the less restrictive condition $m \leqslant (b-1)/3$. Let q_{add} denote the number of grid columns added to describe S_2. In the first case, q_{add}=2(b − 3)+l, where l=1 if $r = 0$, and l=2 otherwise. In the second case, under the condition that $b \geqslant 4$ and thus $b' \geqslant 18$, $n \geqslant 3mb' \geqslant 54m$, we take $q_{\text{add}} = 2\lceil 3m(b-3)/(b-1)\rceil + 1 + \lceil r/(2(b-1))\rceil$. One of the new columns contains $6m$ points of S_2 while the rest at most 2(b − 1) points each. In both cases, the graph G consists of m copies of C_5, $3m$ copies of C_{2b-2} and r isolated vertices.

It can be readily checked that the above claim concerning the reduction from 3DM to QAP remains valid for the modified construction as well.

We end the proof by changing the obtained instance of the QAP slightly and showing that for n_x, n_y, t and w specified in the formulation of the theorem this instance belongs to the set $I(n, n_x, n_y, t, w)$. First we multiply the weight of each edge of G by two. Initially, G is a union of some number of copies of H_4 and, possibly, r isolated vertices (if r defined above is positive). After modification, each copy of H_4 changes into a graph that can be decomposed into 4 copies of H_3 (see Figure 5). The total number of such copies $t' = 24bm - hm$, where h=4 if $b' \equiv 2$ or 3(mod 4) and $h = 16$ otherwise. Since $b \leqslant n/(12m)$ it follows that $t' < 2n$. To reach t specified in the theorem, we can duplicate arbitrary copies of H_3 from those appearing in the decomposition of G (or even use some new copies, for example, consisting of three vertices all belonging to the same chain in G). For each copy of H_3 in the collection obtained we apply the following procedure: take some integer from the interval [1,w] and multiply by it the edge-weights of the copy. The matrix $W(G) = (w_{ij})$ corresponding to the modified graph G is obtained by summing $W(H_3)$ over all copies of H_3 processed by this procedure. To make $W(G)$ nonnegative, we add $w_{\min} := -\min\{w_{ij} | i, j \in N, i < j\}$ to each entry of the upper half of $W(G)$ above the main diagonal.

Now it remains to show that the dimensions of the grid used in the reduction do not exceed n_x and n_y specified in the theorem. Let these dimensions be denoted by n'_x and n'_y. Since each grid row contains at least 9 points of S (exactly 9 if $b' = n/(3m) = 18$) it follows that $n'_y \leqslant n/9$. An upper bound on n'_x can be established by observing that at least $n'_x - 3$ grid columns contain at least 6 points of S each. Thus $n'_x < n/5$ for sufficiently large n. So, $n'_x < n_x$ and $n'_y \leqslant n_y$. To get a grid of size $n_x \times n_y$, we expand the grid obtained by adding $n_x - n'_x$ new columns and $n_y - n'_y$ new rows (these new columns and rows can even be inserted into the grid by shifting some of the existing columns to the right and rows upwards).

It is easy to see that the modified construction is still working in the proof of the reduction, and now a given instance of 3DM has an exact matching if and only if $f_0(W(G), D(S)) = w_{\min}\Sigma(D)$. Moreover, in the case when the answer for 3DM is positive, the instance of the QAP constructed using S and modified graph G belongs to the set $I(n, n_x, n_y, t, w)$. Note that Step 3 of the algorithm GENERATE can be ignored when deciding whether an instance of the QAP is a member of the set I. This step only describes a possible way in which the distance matrix D from the set S is obtained. Since the reduction is polynomial in time the set $I(n, n_x, n_y, t, w)$ is hard. The proof is complete. □

REMARK 2. In the reduction considered above the number of points $n \geqslant 108$. This follows from the restrictions $m \geqslant 2$ and $n \geqslant 54m$.

REMARK 3. In Step 2 of GENERATE, a condition concerning three entries of W defined by a triplet $(i, j; k)$ is checked. This allows to avoid selecting two triplets with corresponding triangles having common edge the weight of which is positive in one of the triangles and negative in another. Releasing this condition leads to a modification of GENERATE for which the set of possible instances includes I and, therefore, is hard too.

Next, we state a theorem concerning asymptotic behaviour of the reduction. We assume that parameters t and w used to define the set I are bounded in the same way as in Theorem 2.

THEOREM 3. *Let ε be any positive number. Then for sufficiently large n there exists a grid of size less than $(1+\varepsilon)n$ for which the set I is hard.*

Proof. Let ε be fixed and let m denote the cardinality of the sets U_1, U_2, U_3 in 3DM as before. We can assume without loss of generality that ε is less than 1. Relate n to m by fixing $n = \lceil 36m^2/\varepsilon \rceil$ and assume that m is sufficiently large to guarantee that

$$\sqrt{n} \geqslant 8\sqrt{\varepsilon}/(1-\varepsilon). \tag{6}$$

We first show that m satisfies the condition for Case 1 in the proof of Theorem 2 to hold, that is, $m \leqslant (b-2)/3$. Since $b > b'/4 - 1 > n/(12m) - 2$ and $(b-2)/3 > n/(36m) - 4/3$ it is sufficient to ascertain that $m \leqslant n/(36m) - 4/3$. Simplifying this inequality we obtain $36m^2 + 48m \leqslant n$. By the choice of n and (6) we can write $36m^2 + 48m \leqslant \varepsilon n + 8\sqrt{\varepsilon n} \leqslant n$. Thus, the condition for Case 1 to hold is established.

Given an instance of 3DM, the corresponding instance of the QAP is constructed using the same reduction as in the previous proof. Since the number of additional columns $q_{\text{add}} \leqslant 2(b-3) + 2 < 2b \leqslant n/(6m)$, the size of the grid obtained is less than $6m(6m + n/(6m)) \leqslant \varepsilon n + n = (1+\varepsilon)n$. The assertion about hardness of the set I defined with respect to this grid follows from the proof of Theorem 2. $\square$

Besides a standard algorithm which can produce each member of I with nonzero probability, we also investigate three other generators each characterized by a set of QAP instances which can be compared to the set I. Two such sets are subsets of I and one is a superset of I. The description of these generators is provided in the next section. We also consider a generator which additionally to triangles selects also both some number of copies of H_5 and some number of copies of H_7. The choice of H_5 and H_7 and not H_4 or H_6 is motivated by a wish to obtain the matrix W (at the beginning of Step 4 of GENERATE) with a smaller value of the ratio $R(W)$ = (sum of positive entries of W)/(-1)(sum of negative entries of W). We can expect that an instance of the QAP with zero optimal value and matrix W having a relatively small sum of positive entries may be harder for QAP heuristics than an instance with the positivity of the matrix W more pronounced.

For a graph G we define the ratio $R(G)$ = (total weight of positive edges of G)$/(-1)$(total weight of negative edges of G). Clearly, $R(H_i) = i/(i-2)$ if i is even, and $R(H_i) = (i+1)/(i-1)$ if i is odd. Particularly, $R(H_4) = 2$, $R(H_5) = R(H_6) = 3/2$ and $R(H_7) = R(H_8) = 4/3$. So, using H_5 and H_7 leads to W with the same value of $R(W)$ as using the same number of H_6 and H_8. Naturally, we prefer graphs H_5 and H_7 since they are smaller and easier to process. We should note that generators using graphs H_i with more than 7 vertices could be constructed. So, an algorithm described below can be considered simply as an illustration of our approach.

In fact, this algorithm is a generalization of GENERATE obtained by inserting two new steps after the initialization step: one for processing h_7 copies of H_7 and another h_5 copies of H_5. The input to the algorithm is that to GENERATE enlarged with the numbers h_7 and h_5. Letting $H_i(\alpha)$, $i = 5$ or 7, denote the graph H_i with weight of each edge multiplied by α, the algorithm can be stated as follows.

```
Algorithm GENERATE_USING_LARGER_GRAPHS
1. Choose $n$ points $s_1, \ldots, s_n$ on the grid $Q(n_x, n_y)$. Set $W$ equal to the all-
   zeros matrix.
2. Repeat $h_7$ times the following operations:
    2.1. Choose a subset $S_7 \subset \{s_1, \ldots, s_n\}$ consisting of 7 points and an
         assignment $p : i \in V \to k$, $s_k \in S_7$, of the vertices of $H_7 =
         (V, E = E_+ \cup E_-)$ to points in $S_7$ such that
           (1) $f(p) = 0$,
           (2) $w_{p(i)p(j)} \geqslant 0$ (if $p(i) < p(j)$) or $w_{p(j)p(i)} \geqslant 0$ (if $p(i) >
               p(j)$) for each $(i, j) \in E_+$, and
               $w_{p(i)p(j)} \leqslant 0$ (if $p(i) < p(j)$) or $w_{p(j)p(i)} \leqslant 0$ (if $p(i) >
               p(j)$) for each $(i, j) \in E_-$,
         allowing possible repetition with some choice (or choices) made
         earlier while executing this step.
    2.2. Choose an integer $\alpha \in [1, w]$.
    2.3. Rename the vertices of $H_7 = (V, E)$ according to $p$. Set $W :=
         W + W(H_7(\alpha))$, where $H_7(\alpha)$ corresponds to $H_7$ just obtained.
3. Repeat $h_5$ times the analogous operations as in Step 2 with respect to the
   graph $H_5$.
4. Perform Steps 2–4 of the algorithm GENERATE.
```

Note that the existence of an assignment p with $f(p) = 0$ can be checked and such an assignment can be obtained using the algorithm BALANCE described in the previous section. This operation and especially checking signs of the entries of W corresponding to the edge set of H_7 or H_5 may significantly restrict the choice of a subset S_7 in Step 2.1 and subset S_5 in Step 3 of the above algorithm.

Let $I'(n, n_x, n_y, h_7, h_5, t, w)$ denote the set of pairs (W, D) which can be produced by GENERATE_USING_LARGER_GRAPHS. As it follows from the next observation this set is not comparable with I.

REMARK 4. If at least one of h_7 and h_5 is positive, then for any n, n_x, n_y, w, t and $t' I'(n, n_x, n_y, h_7, h_5, t', w) \cap I(n, n_x, n_y, t, w) = \emptyset$. To establish this simple fact, it is sufficient to compare the values of R for both algorithms. Thus, before adding $w_{\min}$ to W in GENERATE $R(W)$=2, while at the same point in GENERATE_USING_LARGER_GRAPHS $R(W) < 2$.

The next result is essentially a reformulation of Theorem 2 for the set I'.

THEOREM 4. *For any $n_x \geqslant n/5, n_y \geqslant n/9, t \geqslant 2n$, nonnegative integers h_7, h_5 and positive integer w all bounded from above by some polynomial of n the set $I'(n, n_x, n_y, h_7, h_5, t, w)$ is hard.*

Proof. We need only to notice that matrices $W(H_7(\alpha))$ and $W(H_5(\alpha))$, $\alpha>0$, exist which can be added to W without destroying the reduction used to prove Theorem 2. For example, such a matrix is defined by a graph H_i , i=5 or 7, whose vertices are properly chosen from the vertex set of some copy of the chain C_j, j=5 or $2b-3$ or $2b-2$. 'Chosen properly' means that the restriction of any assignment with zero value for the copy of C_j (see (5) and arguments below) to the vertices of H_i has zero value as well, and, moreover, the edge weights of H_i are in conformance with those of the copy similarly as in the second condition stated in Step 2.1 of GENERATE_USING_LARGER_GRAPHS. □

Note that we can release the second condition in Step 2.1 and its analogues for H_5 and H_3. This does not refute the fact of hardness of the set of possible instances. However, the absence of this condition leads to a matrix W with a larger value of R, what is not desirable.

We end this section with a discussion relating our results with those obtained by Cyganski, Vaz and Virball (1994). They provide a generalization (called a generalized Palubeckis algorithm) of the QAP generator presented in Palubeckis(1988). The algorithm GENERATE described in this paper, in fact, fits into the schema of this generalization. Cyganski, Vaz and Virball(1994) also give a special linear program and prove that this program can be used to compute the optimal solution value for a QAP instance produced by the generalized Palubeckis algorithm. We reformulate their main result using our definitions and notations.

Let $\Theta = \{\Theta_1, \dots, \Theta_r\}$, $r = \binom{n}{2}(n-2)$, be the set of triplets defined with respect to N as before, and $W = (w_{ij})$ be the matrix produced by the algorithm GENERATE. To each triplet $\Theta_l = (i, j; k)$, $l \in \{1, \dots, r\}$, we associate a variable α_l and define $\Theta_l^+ = \{(i,k), (j,k)\}$, $\Theta_l^- = \{(i,j)\}$. Then the linear program given by Cyganski, Vaz and Virball (1994) can be written

$$w_0 = \max w,$$

$$w + \sum_{l,(i,j)\in\Theta_l^-} \alpha_l - \sum_{l(i,j)\in\Theta_l^+} \alpha_l = w_{ij}, i = 1, \dots, n-1, j = i+1, \dots, n, \tag{7}$$

$$\alpha_l \geqslant 0, \quad l = 1, \dots, r.$$

The following statement is a specialization of the theorem proved by Cyganski, Vaz and Virball (1994).

THEOREM 5. *If $W = (w_{ij})$ and D are matrices produced by the algorithm GENERATE, then for the corresponding QAP the optimal value $f_0(W, D)$ is equal* *to*

$$w_0\Sigma(D)/2,$$

where w_0 is the optimal value of the linear program (7). Generally, if the pair of matrices W and D defines an instance of the Euclidean QAP, then $w_0\Sigma(D)/2 \leqslant f_0(W, D)$.

Clearly, if W and D are obtained using algorithm GENERATE, then w_0 equals $w_{\min}$ defined in Step 4 of this algorithm.

In fact, this theorem says that the optimum value of each QAP instance delivered by GENERATE can be computed in polynomial time. Nevertheless, as shown in this section, the set of instances which can be obtained using GENERATE is hard, and no polynomial-time algorithm for the QAP exists, unless P=NP, which can solve each QAP instance in this set. Note that the linear program (7) cannot be used to devise such an algorithm. The job could be done by a linear program for which an analogue of Theorem 5 with $w_0\Sigma(D)/2 \geqslant f_0(W, D)$ substituted for $w_0\Sigma(D)/2 \leqslant f_0(W, D)$ holds. However, the results in this section imply that no such program exists, unless P=NP.

4. Generators

We begin a description of generators with a procedure for which the set of possible instances is a superset of I. Then we describe three implementations of the algorithm GENERATE, which can be ordered according to the strict inclusion relation defined on the corresponding sets of instances that can be produced by these implementations. Finally, we provide some details of a program realizing the algorithm GENERATE_USING_LARGER_GRAPHS.

The input to our first algorithm includes parameters n_x, n_y, t and w defined in the previous section and one additional parameter – an upper bound γ_{bound} on the number of trials in triplet selection. The algorithm can be stated as follows.

```
Algorithm GEN1
1. Form a list $L = (s_1, \dots, s_{n'})$, $n' = n_x n_y$, of grid points by scanning the
   rows of the grid $Q(n_x, n_y)$ sequentially. Set $W$ equal to the all-zeros matrix.
2. Repeat $t$ times the following operations:
    2.1 Set $\gamma := 0$.
    2.2 Randomly select a pair $(i, j) \in \{(k, l) | k, l \in N', k < l, w_{kl} \leqslant 0\}$,
        where $N' = \{1, \dots, n'\}$.
    2.3 Form a set $N'(i, j) = \{l \in N' \setminus \{i, j\} | d(s_i, s_l) + d(s_j, s_l) = d(s_i, s_j)$,
        $w_{il}, w_{li}, w_{jl}, w_{lj}$ are nonnegative$\}$. If $N'(i, j)$ is nonempty, proceed
        to 2.4. Otherwise set $\gamma := \gamma + 1$, and if $\gamma < \gamma_{\text{bound}}$ return to 2.2; else
        stop with a failure.
    2.4 Randomly select $k \in N'(i, j)$ and an integer $\alpha \in [1, w]$. Set $W :=$
        $W + W(T(i, j, k, \alpha))$.
```

3. Eliminate zero rows and columns from W and corresponding points from L, assuming that the correspondence is given by the mapping (ith row and column of W)$\rightarrow s_i$, $i \in N'$. Let $L' = (s'_1, s'_2, \dots, s'_n)$, $n \leqslant n'$, be the list of points upon termination of this procedure.
4. Add $w_{\min} := -\min\{w_{ij} | i, j = 1, \dots, n, i < j\}$ to each entry of W above the main diagonal.
5. Generate random permutation (assignment of objects to points in L') $p = (p(1), \dots, p(n))$.
6. Permute rows and columns of W according to p, i.e., for each pair i, j, $1 \leqslant i < j \leqslant n$, take $w_{ij} = w_{p(i)p(j)}$ if $p(i) < p(j)$ or $w_{ij} = w_{p(j)p(i)}$ otherwise.
7. Construct the distance matrix $D(L')$ by taking d_{ij}, $1 \leqslant i, j \leqslant n, i \neq j$, equal to rectilinear distance between points $s'_i, s'_j \in L'$. Stop with the matrices W and $D = D(L')$ defining a QAP instance, permutation p optimal for this instance and the optimal value $f_0(W, D) = \sum_{ij} w_{ij} d_{p(i)p(j)} = w_{\min} \Sigma(D)/2$.

The algorithm either produces an instance or reports about a failure. The latter, however, is possible only for large t and very small γ_{bound}.

It can be easily seen that the first two steps of the above algorithm are reincarnation of the first two steps of GENERATE. As compared to GENERATE, only Step 3 is essentially new. In fact, the action of Step 3 corresponds to removing isolated vertices from $G(W)$.

Let $I_1(n_x, n_y, t, w)$ denote the set of instances which can be obtained using GEN1 (when GEN1 is terminating properly, i.e., with an instance of the QAP). Clearly, $I_1(n_x, n_y, t, w) \supset I(n, n_x, n_y, t, w)$ for positive t and any $n \geqslant 4$. Note that when an instance in I_1 belongs also to I, permutation p delivered by GEN1 is the same as permutation p defined in Proposition 1. However, when t is sufficiently large and $n{<}n_x n_y$, the probability of producing by GEN1 an instance, which belongs to the set I, is extremely small. In practice, the size of QAP instances created by GEN1 is equal to $n_x n_y$. The next algorithm is a variation of GENERATE.

Algorithm GEN2.

1. Compute grid dimensions n_x, n_y applying the same formulas as in the reduction used to prove Theorem 2.
2. Choose randomly n points $s_1, \dots, s_n$ on the grid $Q(n_x, n_y)$. Set W equal to the all-zeros matrix.
3. Apply Step 2 of GEN1 with n' replaced by n.
4. Apply Steps 3 and 4 of GENERATE.

The values of n_x and n_y can be included into the input to GEN2, and so Step 1 of this algorithm can be dropped. In fact, this modification of GEN2 is an exact implementation of GENERATE. In the next section we refer to this modification as GEN2M. In both cases, for GEN2 and for its modification, the set of instances which can appear in an output is equal to $I(n, n_x, n_y, t, w)$. Before closing the description of GEN2, we should note that the random choice of points in Step

2 and sorting in Step 4 of GEN2 (or, more precisely, in Step 3 of GENERATE) replace Steps 5 and 6 of GEN1.

The following algorithm is another modification of GENERATE. It includes the first step (selection of grid points) of the reduction considered in Section 3. In other words, the distance matrix generated by this algorithm is of the same type as the matrix $D(S)$ defined there. In the description below, m denotes the size of sets in 3DM as before. It is assumed that $m \geqslant 2$. Also, m is bounded from above by $n/54$ for certain values of n or by $n/60$ (see previous section for details).

```
Algorithm GEN3
1. Generate randomly $3m$ triplets describing an instance of 3DM with each
   element of $U$ appearing exactly 3 times.
2. Form a list $L = (s_1, \ldots, s_n)$ of grid points belonging to the set $S$ defined in
   the proof of Theorem 2. Set $W$ equal to the all-zeros matrix.
3. Apply Step 2 of GEN1 with $n'$ replaced by $n$.
4. Apply Steps 4, 5, 6 and 7 of GEN1 with $L'$ replaced by $L$.
```

Let $I_3(n, m, n_x, n_y, t, w)$ denote the set of instances produced by this algorithm. Since the choice of the first subset of S is determined by the set of triplets and the second subset of S is formed deterministically, for any feasible value of m $I_3(n, m, n_x, n_y, t, w)$ is only a subset of $I(n, n_x, n_y, t, w)$. Nevertheless, the set $I_3(n, m, n_x, n_y, t, w)$ includes instances obtained using the reduction from 3DM to QAP and thus is hard, too.

The algorithm GEN3 was extended to fully implement the reduction described in the proof of Theorem 2. In this extension, named GEN4, some of the triangles are chosen deterministically (the number t' of such is specified in the proof) and the rest are chosen randomly.

```
Algorithm GEN4
1. Apply Steps 1 and 2 of GEN3.
2. For each of $t'$ copies of $H_3$ specified in the reduction perform the following:
   randomly select an integer $\alpha \in [1, w]$ and set $W := W + W(T(i, j, k, \alpha))$,
   where $i, j, k$ are the vertices of the copy.
3. Set $t := t - t'$. If $t > 0$, apply Step 2 of GEN1 with $n'$ replaced by $n$.
4. Apply Step 4 of GEN3.
```

Let $I_4(n, m, n_x, n_y, t, w)$, $t \geqslant t'$, be the set of instances produced by GEN4. The following two properties are clear: $I_4(n, m, n_x, n_y, t, w) \subset I_3(n, m, n_x, n_y, t, w)$, and the set $I_4(n, m, n_x, n_y, t, w)$, $t \geqslant t'$, is hard.

We also have implemented a variation of the algorithm GENERATE_USING_LARGER_GRAPHS described in Section 3. This variation, named GEN5, is obtained by replacing Steps 1 and 4 of GENERATE_USING_LARGER_GRAPHS with Step 2 and, respectively, Steps 3 and 4 of GEN2. In Step 2.1, a subset S_7 is chosen randomly. For this subset, an assignment p satisfying the conditions stated in Step 2.1 is searched. If search fails, a new subset S_7 is selected. The number of failures is bounded from above by some parameter χ_7 whose value is included into the input to GEN5. If this bound is reached, GEN5 stops with no instance generated. In this case, h_7 should be decreased or(and) χ_7 increased. The same strategy is used to implement Step 3. The hardness of the set of instances which can be produced by GEN5 is established by Theorem 4.

In our experiments we also tried the algorithm proposed by Palubeckis (1988). The description of this algorithm can also be found in Li and Pardalos (1992) and Pardalos, Rendl and Wolkowicz (1994). In the current paper we call this algorithm GEN0. As already remarked in the preceding section, GEN0 is similar to the algorithm GENERATE described there. In fact, GEN0 forms the matrix W in the same way as GEN1-GEN3 – by applying operations of the same kind as those specified in Steps 2.2-2.4 of GEN1. In GEN0, however, the rules for choosing triplets used to modify W are slightly different (see, for example, Pardalos, Rendl and Wolkowicz (1994) for details).

The last algorithm we tried is the second generator proposed by Li and Pardalos (1992) (and named in their paper as GEN2). The reasons of choosing their second and not the first generator are the following: this generator is less similar to GEN0 than the first one which like GEN0 and also generators suggested in this paper is using triangles in the flow matrix construction, and, most importantly, the results of experimentation presented in Li and Pardalos (1992) show that the second generator produces slightly harder QAP instances than the first one. In our experiments we use a specialization of the second generator by Li and Pardalos(1992) for the rectilinear QAP. Their algorithm is adjusted to generate QAP instances in which the matrix D is composed of the shortest rectilinear distances between pairs of points on the grid. This algorithm, denoted GENLP, is of different type than generators based on the triangle selection.

For instances produced by the second generator by Li and Pardalos (1992) the Gilmore-Lawler bound (Gilmore, 1962; Lawler, 1963) is equal to the optimal value of the objective function. Li, Pardalos, Ramakrishnan and Resende (1994, Corollary 3.1) prove that there is no polynomial-time algorithm to find an optimal permutation for a QAP instance with this property, unless P=NP. However, the answer to the question whether the set of instances which can be generated by GENLP is hard or not is not known. We only expect that this set is hard.

5. Experimental results

The goal of experimentation was to compare, using some heuristic for the QAP, the hardness of QAP instances produced by different generators. As an examiner, the multi-start descent technique was chosen. This algorithm consists of two steps executed intermittently: randomly generating a starting solution, and applying the descent procedure which improves a given solution by performing the pairwise interchanges of objects and stops when no interchange leads to a solution with smaller value of the objective function. The simplest stopping criterion for multi-start descent is the upper bound on the number of repetitions of these two steps. Being simple and easily implementable, the multi-start descent heuristic yields solutions of sufficiently high quality, not much worse than those obtained using more sophisticated methods (Taillard, 1995). If good solutions are needed in a short amount of time, Taillard (1995, beginning of Section 5) recommends to use either methods based on the tabu search, robust (Taillard,1991) or reactive (Battiti and Tecchiolli,1994), or simply multi-start descent procedure.

For evaluation of the quality of solutions, we had to decide which measure to use: the traditional one

$$K'(f_{\text{heu}}) = 100(f_{\text{heu}} - f_0)/f_0,$$

where f_{heu} is a heuristic solution value, or that with the reference to an average value of f

$$K(f_{\text{heu}}) = 100(f_{\text{heu}} - f_0)/(f_{\text{ave}} - f_0),$$

where $f_{\text{ave}} = (\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij})(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij})/\binom{n}{2}$ is the average value of f taken over the set of all permutations. Despite of the fact that the first measure is widely used in combinatorial optimization, we preferred the second one. The reason behind this choice is a sensitivity of $K'(f_{\text{heu}})$ to constant transformations of any of the matrices W and D. In other words, the value $K'(f_{\text{heu}})$ for QAP with each nondiagonal entry of D (or each entry of the upper half of W above the main diagonal) enlarged by a nonzero constant is different from the value $K'(f_{\text{heu}})$ for the initial QAP. It is clear that in both cases the positions of solutions in their ordering according to the values of f are the same. Given a QAP instance (W, D) and a heuristic solution, one can add a sufficiently large positive constant to, say, matrix W and obtain an equivalent QAP for which the value of K' on the same solution is sufficiently close to zero. Thus, different estimates for QAPs with the same structure can be obtained. Obviously, the measure $K(f_{\text{heu}})$ is free from this drawback.

The multi-start descent algorithm and all generators described in this paper have been coded in the C language, and the experiments were done on an IBM PC-486/80.

To obtain the results presented in this section, we run generators with the input values given in Table 1. The rows of this table are partitioned into two clusters: the first containing rows 1 through 8 and the second consisting of the remaining 10 rows. The data in the first cluster were used by all generators, whereas the data in the second only by GEN0, GEN1, GEN2M, GEN5 and GENLP. The first column of this table shows the dimension of the problem. The second and third give the number of triangles (selected to modify the matrix W) for GEN5 and for the rest of generators except GENLP, respectively. In Sections 3 and 4 this characteristic is denoted by t. Next two columns contain the values used (also to modify W) only by GEN5: the number of graphs isomorphic to H_5 and, respectively, to H_7. The rule according to which t_{GEN5}, h_5 and h_7 are calculated is very simple. We fix the number r=3t_{rest} of edges in t_{rest} triangles and use 0.5r, 0.3r and 0.2r edges to define the number of triangles, copies of H_5 and copies of H_7, respectively. Let g_i,

Table 1. Input to generators

n	t_{GEN5}	t_{rest}	h_5	h_7	Set size	Grid dimensions (case of GEN0, GEN1; case of GEN2–GEN5, GEN2M, GENLP)
108	1501	3000	269	86	2	12×9; 17×12
120	1750	3500	315	100	2	12×10; 20×12
132	2250	4500	404	129	2	12×11; 19×12
144	2751	5500	495	157	2	12×12; 22×12
162	3251	6500	584	186	3	18×9; 25×18
180	4249	8500	765	243	3	15×12; 30×18
198	5001	10000	899	286	3	18×11; 29×18
216	5999	12000	1080	343	4	18×12; 33×24
30	101	200	17	6		6×5; 8×8
40	173	350	32	10		8×5; 9×9
50	301	600	54	17		10×5; 10×10
60	451	900	80	26		10×6; 11×11
70	599	1200	109	34		10×7; 12×12
80	801	1600	143	46		10×8; 13×13
90	1001	2000	180	57		10×9; 14×14
100	1250	2500	226	71		10×10; 15×15
120	1750	3500	315	100		12×10; 16×16
150	2751	5500	495	157		15×10; 18×18

$i \geqslant 3$, denote the number of edges of H_i. Clearly, $g_i = i(i-1)/2$, $i \geqslant 3$. Since each of t_{GEN5}, h_5 and h_7 must be an integer, we take t_{GEN5} to be very close but not necessarily equal to $0.5r/g_3 = r/6$, h_5 to $0.3r/g_5 = 0.03r$ and h_7 to $0.2r/g_7 = r/105$. For each n these choices are such that the inequality $|r-(3t_{\text{GEN5}}+10h_5+21h_7)| \leqslant 1$ is satisfied. This strategy guarantees that for GEN5 the number of modifications of the entries of W is almost the same as for generators based on the triangles only. The column under heading ‘set size’ gives the size of sets in 3-dimensional matching. This characteristic is used by GEN2–GEN4. The last column shows the grid dimensions which for GEN0 and GEN1 (the first member of the pair) are different from those for GEN2–GEN5, GEN2M, GENLP (the second member of the pair). In the case of GEN2–GEN4 the grid dimensions do not belong to the input but are computed at the beginning of generation procedure. For the first 8 values of n these computed dimensions were used by GEN5, GEN2M and GENLP. Remember that the reduction from 3DM to QAP can be simulated and, therefore, each of GEN2–GEN4 applied only if $n \geqslant 108$ (see Remark 2, Section 3). For the rest values of n GEN2M, GEN5 and GENLP were run with square grids containing $2n$ or slightly more points. For GEN1, in all cases the number of grid points appeared to be equal to the number of objects. Some parameters are not presented in the table. The value

Table 2. Comparison of generators using multi-start descent (for each generator, minimal values of K over 10 starts in the first row, and averages in the second row)

	Problem size							
Generator	108	120	132	144	162	180	198	216
GEN0	3.5	0.1	1.4	2.1	1.2	2.1	1.7	1.8
	9.0	9.1	7.0	6.8	8.4	7.9	7.3	6.5
GENLP	3.9	2.2	1.6	2.8	0.9	2.0	1.9	4.3
	8.7	5.6	4.5	7.2	8.8	6.6	6.9	10.2
GEN1	5.5	5.4	4.9	4.9	5.6	4.1	3.9	2.7
	12.1	8.9	12.2	9.4	10.4	7.6	7.1	8.3
GEN2	6.3	6.7	5.8	5.4	5.4	5.5	5.2	6.7
	8.2	10.7	11.5	10.1	10.9	10.5	9.3	10.9
GEN3	8.3	4.6	4.7	7.1	7.3	6.1	4.0	6.4
	12.5	10.9	11.0	9.6	10.4	11.8	8.9	12.1
GEN4	7.6	8.9	7.1	6.4	8.5	5.7	5.7	5.5
	13.0	12.7	11.0	11.0	13.5	10.2	10.9	10.1
GEN5	8.0	7.2	5.1	6.0	4.4	6.1	3.7	2.3
	12.3	11.1	13.7	12.4	11.4	12.0	11.4	11.6

Table 3. Number of instances solved optimally (25 instances tried for each entry of the table)

	Problem size; number of starts of descent				
Generator	30; 1000	40; 500	50; 200	60; 100	70; 50
GEN0	25	25	20	22	21
GENLP	25	25	17	14	3
GEN1	23	18	15	10	1
GEN2M	24	19	15	2	2
GEN5	8	8	9	3	1

of w in all the experiments was set to 10. In GEN0, the same value was used as a constant assigned to entries of W. The values of the other parameters were the following: $\gamma_{\text{bound}} = 5$, $\Delta_A = 100$ for GENLP (Δ_A is defined in Li and Pardalos (1992)), $\chi_7 = \chi_5 = 10n$ for GEN5 (χ_5 is the bound on the number of failures in choosing a copy of H_5 and is similar to χ_7 defined in Section 4).

The main results of experiments are summarized in Tables 2–5. To obtain Table 2, the first data cluster in Table 1 was used. For each n available one QAP instance was created by each generator. To solve an instance, the descent procedure was applied to 10 random starting solutions. Sufficiently large computing times did not allow increasing the number of starts. For example, for n=198 10-start descent took more than 2 hours on an IBM PC-486/80 for GEN0–GEN5 and more than 1.5 hours for GENLP. As it can be seen from the table, the best value of K for both GEN0 and GENLP is significantly smaller than for the other generators. Among these GEN4 can be distinguished for which this value is largest in a half of cases.

Table 4. The quality of solutions obtained in 1.5–2.5 hours of computer time (minimal values of K in the first row, and averages in the second row)

	Problem size; number of starts of descent				
Generator	80; 270	90; 170	100; 120	120; 60	150; 30
GEN0	0.3	0.3	0.2	0.1	0.5
	8.7	9.8	8.4	8.7	8.3
GENLP	1.9	1.4	1.4	0.7	4.0
	10.7	9.6	10.2	8.5	9.3
GEN1	2.0	1.7	3.3	4.5	5.3
	12.2	11.6	11.0	10.4	9.2
GEN2M	0	2.1	5.7	2.6	5.8
	13.5*	12.6	11.5	10.2	11.1
GEN5	0.5	1.8	5.4	4.2	6.0
	16.5	14.9	14.8	12.5	13.2

*The average for 144 starts (after reaching an optimal solution).

Table 5. Computational results for problems produced by GEN4 tuned to simulate the reduction from 3DM to QAP (10 starts of descent for each n)

n	t	K_{min}	K_{ave}
108	184	6.8	7.5
120	208	6.6	7.3
132	232	7.1	7.8
144	256	6.5	7.4
162	276	6.2	6.8
180	312	5.3	5.9
198	348	4.5	6.3
216	368	4.4	5.4

Table 3 contains the results of the most interesting, in our opinion, experiment. Multi-start descent was applied to smaller size QAP instances produced by five generators. GEN3 and GEN4 were not run since they are working only for $n \geqslant 108$. GEN2 was represented by its modification GEN2M. The table shows how many instances (out of 25) were solved to optimality in each case defined by the pair (generator; problem size n). The number of starts of descent was limited by computing time resources. Under the given values of the number of starts the worst-case time of solving one instance is about 10 min on our computer for $n = 30$ and about 15 min for the rest n. Here and below, the term ‘worst-case’ is used to characterize situation when multi-start descent fails to reach an optimum. As it follows from Table 3, the hardest QAP instances (for multi-start descent, of course) are generated by GEN5. Even problems of size n=30 are not easily solvable. The results for

GEN2M and GEN1 are quite similar except the case of n=60. As expected, the easiest QAP instances are produced by GEN0.

The results for larger n are displayed in Table 4. The data were taken from the second cluster in Table 1. The number of starts of descent was selected to keep the worst-case solution time between 2 and 2.5 hours on an IBM PC-486/80 for each of GEN0, GEN1, GEN2M and GEN5. For GENLP this time was smaller, about 1.5 hours. In this table, the sums of the minimal values of K for GEN1, GEN2M and GEN5 are similar. GEN0 is again in the last position. Particularly, the results in Tables 2–4 confirm the assertion made by Li and Pardalos (1992) that the QAPs produced by their second generator are harder for QAP heuristics than those generated by the algorithm proposed by Palubeckis (1988), that is, by GEN0.

Table 5 presents computational results for QAP instances produced by GEN4 tuned to mimic the reduction used to prove Theorem 2. The number of triangles t (second column of the table) was taken equal to t' specified in this proof. In this regime of GEN4, an instance of 3DM with a positive answer is constructed randomly, whereas all object locations on the grid and all triangles (but not their weights) are selected deterministically. The values of n and 'set size' were extracted from the first cluster in Table 1. For each n one problem was generated. To each problem 10-start descent was applied. The third and fourth columns of the table list the minimal and, respectively, the average values of K for solutions obtained. We see that the values of $K_{\min}$ in Table 5 are smaller, except two cases, than the corresponding values for GEN4 in Table 2 but are larger than those for some other generators, for example GEN1.

Closing this section we should mention that the results obtained when the measure K' instead of K is used are slightly different. For example, from the analogue of Table 2 prepared using K' it follows that GENLP and GEN0 are the best (the minimal values of K' are largest) leaving GEN4 in the third place. The values of $K'_{\min}$ replacing $K_{\min}$ in Table 5 are 10 times less than the values of K' for GEN4 in the general case. For K, they are comparable.

6. Conclusions

In this paper we defined a set whose members are rectilinear QAP instances, constructed using triangles, with known provably optimal solutions. We have proved that this set is hard, that is, no polynomial-time approximation algorithm for the QAP exists, unless P=NP, which can solve each instance in the set . We have shown that the set remains such for the grid of size (asymptotically) close to the number of objects. Besides these main results, we have established a zero lower bound on the minimal value of the objective function for QAP instances whose flow matrices are defined by special graphs, named PB-graphs, and for one important type of such graphs provided a characterization of point sets on the plane when this bound is tight. As an example, we gave another hard set consisting of QAP instances for construction of which special procedures for making the copies of PB-graphs larger than triangles are applied.

We have described a series of generators with hard output sets of instances with known optimal solutions and compared these and also two existing generators from the literature using the well-known multi-start descent heuristic. It follows from the experiments that with regard to obtaining optimal or close to optimal solutions within a reasonable amount of time the instances produced are hard enough for this heuristic. In fact, for the number of objects about 100, hours of IBM PC-type computer time are required. Based on the results of experimentation, the generator GEN2M can be recommended for its simplicity and sufficient hardness of QAP instances created. At least for smaller n the more difficult problems are obtained using GEN5. We expect that the instances produced by the generators in the series may appear rather difficult for other algorithms for the QAP as well.

Finally, we remark that the PB-graphs H_i , $i \geqslant 5$, can be used not only for instance generation purposes (as in GEN5) but also to improve lower bounds on the optimal value for the QAPs. A lower bounding algorithm of such type is presented in Palubeckis (1997).

Acknowledgments

I wish to thank the anonymous referees for their detailed comments and suggestions that led to this improved version of the paper.

References

Battiti, R. and Tecchiolli, G. (1994), The Reactive Tabu Search, *ORSA J. on Computing* 6, 126–140.

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

Burkard, R.E., Çela, E., Pardalos, P.M. and Pitsoulis L.S. (1998), The Quadratic Assignment Problem, SFB Report 126, Institute of Mathematics, Technical University Graz, Austria, to appear in Pardalos, P.M. and Du, D.-Z. (eds.), *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, Dordrecht/Boston/London.

Burkard, R.E., Karisch, S.E. and Rendl, F. (1997), QAPLIB - A Quadratic Assignment Problem Library, *J. of Global Optimization* 10, 391–403.

Burkard, R.E. and Rendl, F. (1984), A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems, *European J. of Operational Research* 17, 169–174.

Çela, E. (1998), *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht/Boston/London.

Clausen, J. and Perregaard, M. (1997), Solving Large Quadratic Assignment Problems in Parallel, *Computational Optimization and Applications* 8, 111–127.

Connolly, D.T. (1990), An Improved Annealing Scheme for the QAP, *European J. of Operational Research* 46, 93–100.

Cyganski, D., Vaz, R.F. and Virball, V.G. (1994), Quadratic Assignment Problems Generated with the Palubetskis Algorithm Are Degenerate, *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications* 41, 481–484.

Dyer, M.E. and Frieze, A.M. (1985), On the Complexity of Partitioning Graphs into Connected Subgraphs, *Discrete Applied Mathematics* 10, 139–153.

Feo, T.A. and Resende, M.G.C. (1995), Greedy Randomized Adaptive Search Procedures, *J. of Global Optimization* 6, 109–133.

Finke, G., Burkard, R.E. and Rendl, F. (1987), Quadratic Assignment Problems, *Annals of Discrete Mathematics* 31, 61–82.

Fleurent, C. and Ferland, J.A. (1994), Genetic Hybrids for the Quadratic Assignment Problem, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 16, 173–187.

Gambardella, L.M., Taillard, E.D. and Dorigo, M. (1997), Ant Colonies for the QAP, Technical Report IDSIA-4-97, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano, Switzerland.

Garey, M.R. and Johnson, D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H.Freeman and Co., San Francisco.

Gilmore, P.C. (1962), Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem, *J. SIAM* 10, 305–313.

Hasselberg, J., Pardalos, P.M. and Vairaktarakis, G. (1993), Test Case Generators and Computational Results for the Maximum Clique Problem, *J. of Global Optimization* 3, 463–482.

Lawler, E.L. (1963), The Quadratic Assignment Problem, *Management Science* 9, 586–599.

Li, Y. and Pardalos, P.M. (1992), Generating Quadratic Assignment Test Problems with Known Optimal Permutations, *Computational Optimization and Applications* 1, 163–184.

Li, Y., Pardalos, P.M., Ramakrishnan, K.G. and Resende, M.G.C. (1994), Lower Bounds for the Quadratic Assignment Problem, *Annals of Operations Research* 50, 387–411.

Li, Y., Pardalos, P.M. and Resende, M.G.C. (1994), A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 16, 237–261.

Mautor, T. and Roucairol, C. (1994), A New Exact Algorithm for the Solution of Quadratic Assignment Problems, *Discrete Applied Mathematics* 55, 281–293.

Murthy, K.A., Pardalos, P.M. and Li, Y. (1992), A Local Search Algorithm for the Quadratic Assignment Problem, *Informatica* 3, 524–538.

Palubeckis, G.S. (1988), A Generator of Quadratic Assignment Test Problems with Known Optimal Solution, *U.S.S.R. Computational Mathematics and Mathematical Physics* 28, 97–98. [Translated from *Zh. Vychisl. Mat. Mat. Fiz.* 28, 1740–1743.]

Palubeckis, G. (1997), The Use of Special Graphs for Obtaining Lower Bounds in the Geometric Quadratic Assignment Problem, *Informatica* 8, 377–400.

Pardalos, P.M., Rendl, F. and Wolkowicz, H. (1994), The Quadratic Assignment Problem: A Survey and Recent Developments, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 16, 1–42.

Pardalos, P.M., Resende, M.G.C. and Li, Y. (1996), FORTRAN Subroutines for Approximate Solution of Dense Quadratic Assignment Problems Using GRASP, *ACM Transactions on Mathematical Software* 22, 104–118.

Pardalos, P.M., Resende, M.G.C. and Pitsoulis, L.S. (1997), Algorithm 769: FORTRAN Subroutines for Approximate Solution of Sparse Quadratic Assignment Problems Using GRASP, *ACM Transactions on Mathematical Software* 23, 196–208.

Sanchis, L.A. (1990), On the Complexity of Test Case Generation for NP-Hard Problems, *Information Processing Letters* 36, 135–140.

Sanchis, L.A. and Jagota, A. (1996), Some Experimental and Theoretical Results on Test Case Generators for the Maximum Clique Problem, *INFORMS J. on Computing* 8, 87–102.

Skorin-Kapov, J. (1990), Tabu Search Applied to the Quadratic Assignment Problem, *ORSA J. on Computing* 2, 33–45.

Taillard, E. (1991), Robust Taboo Search for the Quadratic Assignment Problem, *Parallel Computing* 17, 443–455.

Taillard, E.D. (1995), Comparison of Iterative Searches for the Quadratic Assignment Problem, *Location Science* 3, 87–105.

Tate, D.M. and Smith, A.E. (1995), A Genetic Approach to the Quadratic Assignment Problem, *Computers and Operations Research* 22, 73–83.

Wilhelm, M.R. and Ward, T.L. (1987), Solving Quadratic Assignment Problems by Simulated Annealing, *IEEE Transactions* 19, 107–119.