INFORMATICA, 2000, Vol. 11, No. 3, 281–296

An Algorithm for Construction of Test Cases for the Quadratic Assignment Problem

Gintaras PALUBECKIS
Department of Practical Informatics, Kaunas University of Technology
Studentų 50, 3031 Kaunas, Lithuania
e-mail: gintaras@soften.ktu.lt

Received: March 2000

Abstract. In this paper we present an algorithm for generating quadratic assignment problem (QAP) instances with known provably optimal solution. The flow matrix of such instances is constructed from the matrices corresponding to special graphs whose size may reach the dimension of the problem. In this respect, the algorithm generalizes some existing algorithms based on the iterative selection of triangles only. The set of instances which can be produced by the algorithm is NP-hard. Using multi-start descent heuristic for the QAP, we compare experimentally such test cases against those created by several existing generators and against Nugent-type problems from the QAPLIB as well.

Key words: combinatorial optimization, quadratic assignment problem, test cases.

1. Introduction

The *quadratic assignment problem* (QAP) can be stated as follows. Given a finite set $N = \{1, 2, \ldots, n\}$ and three $n \times n$ matrices $W = (w_{ij})$, $D = (d_{ij})$ and $A = (a_{ij})$ with real entries, find a permutation p of the set N that minimizes

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

In this paper we deal with the instances of this problem in which the matrix D is composed of the shortest rectilinear distances between pairs of n points in the plane. This partial case and its generalizations to higher dimensions as well is called *rectilinear QAP*.The QAP serves as a mathematical model for many problems coming from real-world applications. In the framework of the facility location problem, there are n facilities that have to be assigned to n different locations. The matrix D describes the distances between locations, the matrix W represents the flows between facilities, and the matrix A gives the fixed costs of the assignment of facilities to locations. The flow between facility i and facility j assigned to locations k and l, respectively, contributes a cost equal to the product $w_{ij}d_{kl}$. The fixed cost of assigning facility i to location k is given by the entry a_{ik} of the matrix A. The objective is to find an assignment of all facilities to all locations,

such that the total cost of the assignment is minimized. Other applications of the QAP are described by Geoffrion and Graves (1976), Elshafei (1977), Krarup and Pruzan (1978), Burkard (1984), Hubert (1987), Laporte and Mercure (1988). The most important research results on the QAP are reviewed in the recent book by Çela (1998) and in the survey papers by Pardalos, Rendl and Wolkowicz (1994), and Burkard *et al*. (1998).

The QAP is generally recognized as a very difficult combinatorial optimization problem. The exact algorithms (see, for example, Mautor and Roucairol, 1994; Clausen and Perregaard, 1997) can solve in a reasonable amount of time only small instances of the QAP – with the number of objects less than 30. So, the heuristic algorithms are widely used, including simulated annealing (Burkard and Rendl, 1984; Wilhelm and Ward, 1987; Connolly, 1990), 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), genetic (Fleurent and Ferland, 1994; Tate and Smith, 1995), scatter search (Cung *et al*., 1996) and ant system (Gambardella, Taillard and Dorigo, 1997) techniques among others. Usually, these algorithms are tested on QAP instances, including larger benchmark problems in the QAPLIB (Burkard, Karisch and Rendl, 1997), for which the optimal solutions are not known. The test cases of another kind are those produced by special generators that together with the matrices defining an instance of the QAP provide us with a provably optimal solution for this instance. For the rectilinear QAP such generators are proposed by Palubeckis (1988, 1999), and for arbitrary (not necessarily rectilinear and even Euclidean) QAP by Li and Pardalos (1992). The results of computational experiments (Li and Pardalos, 1992; Palubeckis, 1999) show that the test cases constructed by these generators are rather hard for several heuristics for the QAP, namely, simulated annealing (Burkard and Rendl, 1984), tabu search (Skorin-Kapov, 1990), graph-partitioning (Murthy, Pardalos and Li, 1992) and multi-start descent.

For a combinatorial optimization problem, the hardness of the set of instances can be defined formally (see Sanchis, 1990).

DEFINITION 1. (Sanchis, 1990). 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.

In Palubeckis (1999) a series of QAP instance generators is described. For each of them the set of possible outputs (matrices W, D) is hard (in a sense of the definition given above with QAP in the role of Π). According to theorem due to Cyganski, Vaz and Virball (1994) the optimal value for instances produced by these generators as well as by an earlier algorithm of Palubeckis (1988) can be revealed in polynomial time. Nevertheless, the existence of polynomial-time algorithm for solving such instances to optimality is impossible, unless P=NP.

In this paper we present an algorithm for constructing test cases for the QAP, which can be viewed as a generalization of the generators proposed by Palubeckis (1988, 1999). The algorithm differs from those generators mainly in that it uses larger graphs in the

construction of the flow matrix – of size up to the dimension of the problem. The set of QAP instances that can be produced by this algorithm is hard. Using multi-start descent heuristic for the QAP, we compare the instances of size $n = 20, 22, 24$ and 30 created by our algorithm with four benchmark problems procured from the QAPLIB (Burkard, Karisch and Rendl, 1997): Nug20 and Nug30 due to Nugent, Vollmann and Ruml (1968), and also Nug22 and Nug24 constructed out of Nug30 by deleting certain rows and columns (Clausen *et al*., 1996). All these problems are instances of the rectilinear version of the QAP, too. Computational results show that for multi-start descent the test cases produced by our algorithm are harder than Nugent-type problems of the same size. We also present the results of comparison of this algorithm with several existing generators (Palubeckis, 1988, 1999; Li and Pardalos, 1992).

The paper is organized as follows. In Section 2 we give a description of the algorithm. In Section 3 we report computational results. Finally, in Section 4 we make some concluding remarks.

We end the introduction with some basic definitions and notations. We will write $D(B)$ to denote the rectilinear distance matrix defined by the set $B = \{b_1, \ldots, b_n\}$ of points on the plane. For $b_i = (x_i, y_i), b_j = (x_j, y_j) \in B$ the corresponding entry of $D(B)$ is equal to $|x_i - x_j| + |y_i - y_j|$. Since D in (1) is the distance matrix, we can assume without loss of generality that the lower half of W under the main diagonal is zero, that is, $w_{ji} = 0$ for all i, j such that $i < j$ (if $w_{ji} \neq 0$ for some pair $i, j, i < j$, we can set $w_{ij} := w_{ij} + w_{ji}, w_{ji} := 0$). We can also assume that the main diagonal of W is zero because such is the main diagonal of D.

We denote a graph by $G = (V, E)$ where V is the set of vertices and E is the set of edges. If the edges are oriented we have a *digraph*. The *size* of the graph (digraph) is the number of vertices in V. To any graph $G = (V, E), V \subseteq N$, with edge weights $c_{ij}, (i, j) \in E$, we can associate an $n \times n$ flow matrix $W(G)$ with the following entries: $w_{ij} = c_{ij}$ (if $i < j$) or $w_{ji} = c_{ij}$ (if $i > j$) if $(i, j) \in E$, and $w_{ij} = 0$ otherwise. We denote by $W(G, \alpha), \alpha \neq 0$, the flow matrix obtained from $W(G)$ by multiplying each entry of $W(G)$ by α.

Given integers $n_x \geqslant 1, n_y \geqslant 1$, we define a *regular* $n_x \times n_y$ *grid* $Q(n_x, n_y)$ as a set $\{(i, j) \mid i = 1, \ldots, n_x, j = 1, \ldots, n_y\}$ of points on the plane.

In this paper we restrict our attention to QAP instances without a linear part, that is, to (1) with zero matrix A. We denote the optimal value of (1) with zero A by $f_0(W, D)$.

2. The Algorithm

Before giving a description of the algorithm, we will specify graphs which play the main role in the construction of the flow matrix, one of two defining an instance of the QAP. They are members of the following class of graphs introduced by Palubeckis (1997).

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 $R(G, V')$ of the weights in the

set $\{c_{ij} \mid (i,j) \in E, i \in V', j \in V \setminus V'$ or $i \in V \setminus V', j \in V'\}$ is nonnegative for each subset $V' \subset V$.

In Palubeckis (1997) the following fact is established.

Lemma 1. *If W is such that $G(W)$ is a PB-graph, then for any rectilinear distance matrix D the optimal value $f_0(W, D)$ of the corresponding QAP is nonnegative.*

An important type of PB-graphs, in which we are interested here, can be defined as follows. For $m \geqslant 3$, the vertex set V of the graph consists of two disjoint subsets V_m^1 and V_m^2 of sizes $\lceil m/2 \rceil$ and $\lfloor m/2 \rfloor$, respectively. The edge set E consists of the subset $E_+ = \{(i,j) \mid i \in V_m^1, j \in V_m^2\}$ of positive edges and the subset $E_- = \{(i,j) \mid i,j \in V_m^1$ or $i,j \in V_m^2\}$ of negative edges. So, the graph is a complete signed graph. We denote this graph by $H_m = (V, E)$.

The algorithm we present in this paper is based on the general schema developed by Palubeckis (1999). The distance matrix D is formed by selecting a prescribed number of grid points and calculating shortest distances between them. The flow matrix W is constructed by summing up some number of matrices each of which, except possibly one, is defined by a copy of $H_m, m \in [3, n]$, with edge weights multiplied by some positive scalar. The last matrix corresponds to a complete graph. We need it to ensure the nonnegativity of W.

Presentation of the algorithm is organized in a top-down manner. We give a chain of three procedures where each of the last two is invoked by its predecessor. The input to the main procedure includes the number of objects n, grid dimensions n_x, n_y, the number of graphs to be selected h, lower and upper bounds on the size of graphs $m_{\text{lower}} \geqslant 3, m_{\text{upper}} \leqslant n$, upper bound $w \geqslant 1$ on the weight of edges, and an upper bound ρ_{bound} on the number of trials in graph selection. The algorithm can be stated as follows.

Algorithm GENERATE_INSTANCE

1. Select randomly n points $b_1, \ldots, b_n$ on the grid $Q(n_x, n_y)$. Set W equal to the all-zeros matrix.
2. Repeat h times the following operations:
   2.1. Set $\rho := 0$.
   2.2. Choose randomly an odd number $m \in [m_{\text{lower}}, m_{\text{upper}}]$.
   2.3. Randomly select m points in $B = \{b_i \mid i \in N = \{1, \ldots, n\}\}$. Let $M \subset N, |M| = m$, be the index set specifying these points.
   2.4. Apply procedure BICOLORING. If it returns "true", then proceed to 2.5. Otherwise set $\rho := \rho + 1$, and if $\rho < \rho_{\text{bound}}$ go to 2.3; else stop with a failure.
   2.5. Randomly choose an integer $\alpha \in [1, w]$. Set $W := W + W(G, \alpha)$, where G is a copy of H_m with V_m^1 (respectively, V_m^2) consisting of those elements of the set M which received red (respectively, blue) color in BICOLORING.

3. Sort the points $b_i, i \in N$, in the order of appearance while scanning the rows of the grid $Q(n_x, n_y)$ sequentially. Construct the distance matrix $D(B)$ in such a way that for any $i \in N$ the ith row and column of $D(B)$ would correspond to the ith point in the order obtained.
4. If $w_{\min} := \min\{w_{ij} \mid i, j \in N, i < j\} < 0$, then add $-w_{\min}$ to each entry of W above the main diagonal. Stop with the matrices $W, D = D(B)$, optimal value $w_{\min} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}/2$ and the optimal permutation $p = (l_1, \ldots, l_n)$, where $l_i, i \in N$, is the rank of the point b_i in the sequence obtained in 3.

The goal of the procedure BICOLORING is to construct special bipartite digraph and to obtain some feasible coloring of its arcs with two colors, red and blue. The vertices of the digraph correspond to grid lines, vertical and horizontal, containing at least one point selected in Step 2.3. The arcs are in one-to-one correspondence with the elements of the set $\{b_i \mid i \in M\}$. Each bicoloring, in which red color is used $\lceil m/2 \rceil$ times, defines a copy of H_m with the division of its vertex set $V = M$ into two parts V_m^1, V_m^2. Only some bicolorings are feasible for our purposes, namely those for which a QAP defined by $W(H_m)$ and by $D(B)$ with ith row and ith column, $i = 1, \ldots, n$, corresponding to b_i has the following property: $f_0(W(H_m), D(B)) = 0 = f(p^*)$, where p^* is the identity permutation $(1, 2, \ldots, n)$. For some arrangements of points $b_i, i \in M$, feasible bicolorings do not exist at all. In this case the procedure returns "false". To check feasibility of a bicoloring we use special function β defined on the vertex set of one of the parts. The outlined procedure is now stated formally.

Procedure BICOLORING

1. Form the following sets: set of vertices of the first part $V_1 = \{v(z_i), i = 1, \ldots, q \mid$ for each $i \in \{1, \ldots, q\}$ $z_i = x_k$ for some point $b_k = (x_k, y_k), k \in M$, and for each $k \in M$ $x_k = z_i$ for some $i \in \{1, \ldots, q\}\}$, set of vertices of the second part $V_2 = \{u(z_j), j = 1, \ldots, r \mid$ for each $j \in \{1, \ldots, r\}$ $z_j = y_k$ for some point $b_k = (x_k, y_k), k \in M$, and for each $k \in M$ $y_k = z_j$ for some $j \in \{1, \ldots, r\}\}$, set of edges $E = \{(v(z_i), u(z_j)) \mid b_k = (x_k = z_i, y_k = z_j), k \in M\}$.
2. Set flag $:= 0$. For $j = 1, \ldots, r$ do:
   2.1. If the degree $\delta(u(z_j))$ of the vertex $u(z_j)$ is even, then take $\mu = 0$. Otherwise if flag $= 1$, take $\mu = 1$; else take $\mu = -1$. Set $s := (\delta(u(z_j)) + \mu)/2$. In addition, if $\delta(u(z_j))$ is odd, then set flag $:= 1 -$ flag.
   2.2. Orient randomly selected s edges incident with $u(z_j)$ from a vertex in V_1 to $u(z_j)$, whereas for the rest edges incident with $u(z_j)$ choose the opposite direction.
3. Set flag $:= 1$. For $i = 1, \ldots, q$ do:
   3.1. Calculate the number $\delta_{\text{in}}(v(z_i))$ of incoming and the number $\delta_{\text{out}}(v(z_i))$ of outcoming arcs incident with $v(z_i)$. Set $\Delta_x(i) := \delta_{\text{in}}(v(z_i)) - \delta_{\text{out}}(v(z_i))$.

3.2. If $\delta(v(z_i)) := \delta_{\text{in}}(v(z_i)) + \delta_{\text{out}}(v(z_i))$ is even, then take $\mu = 0$. Otherwise if $\text{flag} = 1$, take $\mu = 1$; else take $\mu = -1$. Set $\beta(i) := \Delta_x(i) - \mu$. In addition, if $\delta(v(z_i))$ is odd, then set $\text{flag} := 1 - \text{flag}$.

4. Apply procedure BALANCE. If it returns "true", then assign red color to elements of M corresponding to arcs oriented from V_2 to V_1 and assign blue to the remaining elements of M (a correspondence between arcs and elements of M is given by the set E constructed in 1). Return the same value as that returned by BALANCE.

To obtain a feasible copy of H_m that can be used in the flow matrix formation, we need to change orientation of some of arcs to make all values of the function β equal to zero. For this, paths from vertices with negative value of β to those with positive value of β are searched. This job is done by the following loop of three steps.

Procedure BALANCE

1. Check whether the set $\{v(z_i) \mid v(z_i) \in V_1 \text{ and } \beta(i) < 0\}$ is nonempty. If so, take any vertex $v(z_l)$ from this set and go to 2. Otherwise return "true".
2. Try to find a path in the digraph constructed in the invoking procedure between $v(z_l)$ and any vertex in V_1 with positive value of β. If failure, then return "false". Otherwise proceed to 3.
3. Change an orientation of each arc on the path found to the opposite. Correct the values of the function β appropriately. Go to 1.

The optimal value for the QAP defined by the matrices $D(B)$ and $W(H_m, \alpha), m$ odd, is equal to zero if and only if procedure BALANCE returns "true", or equivalently, $\beta(i) = 0$ for all $i \in \{1, \ldots, q\}$ is achieved. This fact is a part of Theorem 1 of Palubeckis (1999), which we reformulate here using our current definitions and notations. We assume everywhere below that the flow matrix $W(H_m)$ is of the same dimensions as the corresponding distance matrix.

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

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

for m odd, the algorithm BALANCE called in Step 4 of BICOLORING returns "true".

We apply this theorem for the point set $S = \{b_i \mid i \in M\}$, where M is defined in Step 2.3 of GENERATE_INSTANCE. A necessary and sufficient condition for a permutation p to be optimal and satisfy $f(p) = 0$ for a QAP defined by the pair $W(H_m), D(S)$ is given by Lemma 3 of Palubeckis (1999). Lemma 2 below is a restriction of this lemma to odd m – only such values of m are used in the algorithm described above. Assume

for simplicity that $M = \{1, \ldots, m\}$. Given a permutation p which assigns vertices of $M = V_m^1 \cup V_m^2$ to points in $S = \{b_i \mid i \in M\}$, we can construct a digraph $G(p)$ by taking the graph $(V_1 \cup V_2, E)$ defined in Step 1 of BICOLORING and orienting its edges according to the following rule: from $v(z_i)$ to $u(z_j)$ for $b_k = (x_k = z_i, y_k = z_j)$ if vertex v satisfying $p(v) = k$ belongs to V_m^2, and from $u(z_j)$ to $v(z_i)$ for the same b_k if such a vertex v belongs to V_m^1. Let $J(X)$ be the set of indices $i \in \{1, \ldots, q\}$ for which $\delta(v(z_i))$ (see Step 3.2 of BICOLORING) is odd. Assuming $J(X) = \{i_1, \ldots, i_\gamma\}$, define $J_1(X) = \{i_j \mid i_j \in J(X), j \text{ is odd}\}$, $J_2(X) = J(X) \setminus J_1(X)$. Suppose that $\Delta_x(i)$ is the difference between $\delta_{\text{in}}(v(z_i))$ and $\delta_{\text{out}}(v(z_i))$ as before. Let $J(Y)$, $J_1(Y)$, $J_2(Y)$ and $\Delta_y(j)$ be the analogous sets and difference with respect to the y-coordinate.

Lemma 2. *Let S be a set of an odd number $m \geqslant 3$ points on the plane. A permutation p described by the digraph $G(p)$ is optimal and $f_0(W(H_m), D(S)) = 0$ if and only if:*

$$\Delta_x(i) = \begin{cases} 1, & \text{if} \quad i \in J_1(X), \\ -1, & \text{if} \quad i \in J_2(X), \\ 0, & \text{if} \quad i \in \{1, \ldots, q\} \setminus J(X), \end{cases} \tag{2}$$

$$\Delta_y(j) = \begin{cases} -1, & \text{if} \quad j \in J_1(Y), \\ 1, & \text{if} \quad j \in J_2(Y), \\ 0, & \text{if} \quad j \in \{1, \ldots, r\} \setminus J(Y). \end{cases} \tag{3}$$

Suppose the matrix $D(B)$ is constructed in such a way that for each $i = 1, \ldots, n$ point b_i is represented by the ith row (and column) of it. Consider the QAP defined by $D(B)$ and $W(H_m)$. If BALANCE returns "true", then any permutation p satisfying the following condition is optimal: $p(i) \in V_m^1$ if $i \in V_m^1$, and $p(i) \in V_m^2$ if $i \in V_m^2$, where V_m^1, V_m^2 are subsets of M delivered by BICOLORING. This condition assures that $G(p')$ for p' being a restriction of p to elements of the set M coincides with the digraph at the end of BICOLORING, and thus (2) and (3) in Lemma 2 for p' hold establishing the optimality of p' for the restricted (to points in S) QAP, what is tantamount to the optimality of p for the full QAP. Particularly, the above condition is satisfied by the identity permutation $p^* = (1, 2, \ldots, n)$. This permutation remains optimal (with $f(p^*) = 0$, of course) also for the same matrix $D(B)$ and for W obtained by summing up all h matrices $W(H_m, \alpha)$, that is, at the beginning of Step 3 of GENERATE_INSTANCE. In the case of an arbitrary mapping of points in $B = \{b_1, \ldots, b_n\}$ to rows (columns) of $D(B)$ the rule "ith object goes to point b_i" is apt. Application of this rule gives the optimal permutation $p = (p(1), \ldots, p(n))$, where $p(i), i \in \{1, \ldots, n\}$, is the index of the row (also, column) representing the point b_i. For $D(B)$ constructed in Step 3 of GENERATE_INSTANCE $p(i)$ is equal to l_i defined in Step 4 of the same algorithm. Because points $b_i, i = 1, \ldots, n$, in Step 1 of the algorithm are selected randomly, permutation $(l_1, \ldots, l_n)$ can be considered as being random.

The algorithm just described uses graphs of only odd size. The reason behind such a choice is too small probability of selecting m points $b_i, i = 1, \ldots, m$, m even, for which

the equality $f_0(W(H_m), D(B)) = 0$ would be true. In fact, for m even a condition for the optimal value to be zero is simpler than for odd m (see Theorem 1), but, as our experiments show, rarely satisfied by a point set selected randomly. So, the requirement for m to be odd in Step 2.2 of GENERATE_INSTANCE significantly reduces the time taken by instance construction. Moreover, for larger m_{upper} and smaller ρ_{bound} this restriction allows to avoid terminating of the algorithm with a failure at Step 2.4. On the other hand, we believe that using only odd size graphs does not detract from algorithm's ability to produce test cases which are hard for quadratic assignment heuristics.

EXAMPLE 1. We will illustrate application of the procedures BICOLORING and BALANCE for the set B of the following 11 points on an 8×8 grid (see Fig. 1a): (1,1), (1,3), (1,6), (2,8), (4,4), (5,6), (6,4), (6,5), (6,7), (8,1) and (8,2) (in Fig. 1a each point is marked by an index giving its position in the set B). We have vertices $v(1), v(2), v(4), v(5), v(6), v(8), u(1), u(2), \ldots, u(8)$ and arcs $e_1 = (v(1), u(1)), e_2 = (u(1), v(8)), e_3 = (u(2), v(8)), e_4 = (v(1), u(3)), e_5 = (v(4), u(4)), e_6 = (u(4), v(6)), e_7 = (u(5), v(6)), e_8 = (v(1), u(6)), e_9 = (u(6), v(5)), e_{10} = (v(6), u(7)), e_{11} = (u(8), v(2))$ (their orientation is chosen in Step 2 of BICOLORING). The digraph is shown in Fig. 1b. The initial function β has the following values: $-4, 2, -2, 2, 0, 2$. Since this function is not everywhere zero, the procedure BALANCE is invoked. Starting from $v(1)$ it finds the following two paths: $v(1), u(1), v(8)$ and $v(1), u(6), v(5)$. After replacing arcs e_1, e_2, e_8 and e_9 by $(u(1), v(1)), (v(8), u(1)), (u(6), v(1))$ and $(v(5), u(6))$, respectively, we have $\beta = (0, 2, -2, 0, 0, 0)$. However, no path between $v(4)$ and $v(2)$ exists, so the procedure BICOLORING returns "false". If we modify the given set of points slightly by substituting the point (6,7) with the point (6,8), then the return value would be "true". Indeed, in this case BALANCE finds the path $v(4), u(4), v(6), u(8), v(2)$, and BICOLORING constructs a copy of H_{11} with $V_{11}^1 = \{1, 3, 5, 8, 9, 11\}$ and $V_{11}^2 = \{2, 4, 6, 7, 10\}$.

Let $I(n, n_x, n_y, h, w)$ denote the set of QAP instances, that is, pairs W, D, which can be produced by GENERATE_INSTANCE applied with $m_{\text{lower}} = 3$.

The following result is implied by Theorem 3 of Palubeckis (1999) (since the set considered in its formulation is only a subset of the set $I(n, n_x, n_y, h, w)$ defined above).

Theorem 2. *Let ϵ be any positive number. Then for sufficiently large n there exists a grid $Q(n_x, n_y)$ of size $n_x n_y < (1 + \epsilon)n$ such that for any $h \geqslant 2n$ and $w \geqslant 1$ both bounded from above by some polynomial of n the set $I(n, n_x, n_y, h, w)$ is hard.*

3. Computational Experiments

In this section we present the main results of experimentation with the well-known multi-start descent heuristic for the QAP on the test cases produced by the algorithm we have described here and also by some of the existing generators. Using multi-start descent we

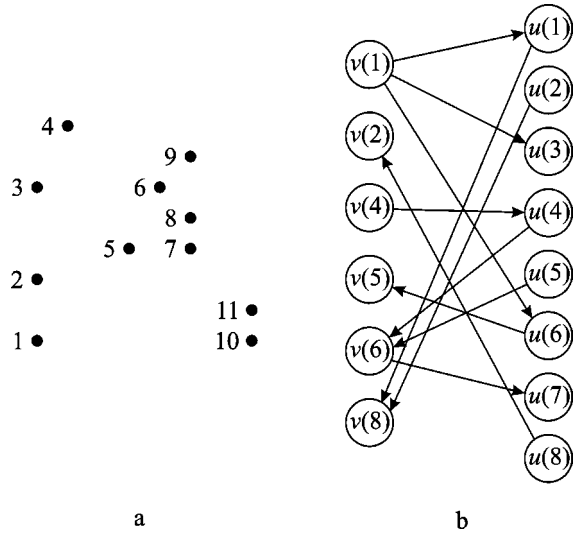

Fig. 1. Example of the directed graph: a – set of points; b – digraph constructed in Steps 1 and 2 of BICOLORING.

tried to evaluate experimentally the hardness of instances constructed by different algorithms. For comparison, we also run this heuristic on two largest quadratic assignment problems due to Nugent, Vollmann and Ruml (1968), named Nug20 and Nug30, and two problems, Nug22 and Nug24, constructed out of Nug30 by deleting certain rows and columns. We have selected these problems from the QAPLIB for several reasons. Firstly, they, especially Nug20 and Nug30, are well known and widely used in experimental evaluation of different algorithms for the QAP. Secondly, all of them except the largest one are solved to optimality, and for the largest problem it is believed that the best known value is optimal. Thirdly, these benchmark problems are instances of the rectilinear QAP, exactly as instances generated by our algorithm, what is desirable to make a comparison more fair.

The multi-start descent heuristic consists of the following two steps executed intermittently.

1. Randomly generate some permutation $p \in P$, where P is the set of all permutations of N.

2. Apply procedure DESCENT(p). If permutation returned by this procedure is better than the best one found so far, then store it as p_{best}. Depending on some termination criterion, either stop with p_{best}, or go to 1.

In our experiments we adopted the simplest termination criterion – an upper bound on the number of starts of descent.

Finding a locally optimal solution used in Step 2 of the above heuristic takes the following steps.

Procedure DESCENT(p_{init})

1. Set $p := p_{\text{init}}$.
2. Search the neighborhood of p defined as $L(p) = \{p' \in P |$ there exist $j, k \in N, j \neq k$, such that $p'(j) = p(k), p'(k) = p(j)$, and $p'(i) = p(i)$ for all $i \in N \setminus \{j, k\}\}$. If $p' \in L(p)$ is found for which $f(p') < f(p)$, then set $p := p'$ and repeat 2. Otherwise, return p.

The multi-start descent is a simple iterative heuristic for the QAP, which, as remarked, for example, by Taillard (1995), produces rather good solutions in a short amount of time, not much worse than solutions yielded by more sophisticated and time-consuming algorithms, like genetic or tabu search.

Besides the algorithm presented in the prior section, in the experiments we also used the following generators:

1. A generator proposed by Palubeckis (1988) (its description can also be found in Li and Pardalos (1992), Pardalos, Rendl and Wolkowicz (1994), and Çela (1998)). We will use acronym GEN0 for it in Tables 3 and 4 below and in the rest of this section.

2. The second generator presented by Li and Pardalos (1992), which we adopt for generating QAP instances with the rectilinear distance matrix. This generator (referred to as GENLP below) is different from the first one of the same authors and from GEN0, since it does not use triangles (graphs H_3) in the construction of the flow matrix.

3. GEN2M from Palubeckis (1999). This generator is kindred to GEN0 – exploits only triangles. Nevertheless, as it is proved by Palubeckis (1999) the set of instances produced by GEN2M is hard. In the current paper this generator will be denoted as GEN1.

4. GEN5 from Palubeckis (1999) – a generator which together with H_3 uses also H_5 and H_7. While constructing the flow matrix W the positive edge weights of the graphs are added only to nonnegative entries of W, and negative only to nonpositive entries of W. The output set is hard, too. Here we refer to this generator as GEN2.

To evaluate the quality of solutions we use the following measure

$$K(f_h) = 100(f_h - f_0)/(f_a - f_0),$$

where f_h is a heuristic solution value, and f_a is the average value of f taken over the set P

$$f_a = (\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij})(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij})/(n(n-1)/2).$$

Our choice in favor of this measure against the simpler one $K'(f_h) = 100(f_h - f_0)/f_0$ is motivated by the fact that the latter is sensitive to transformations of any of two matrices when an arbitrary constant is added to each entry of the matrix W (or D). Clearly, such transformations do not affect the value of $K(f_h)$.

All the programs implementing algorithms described or mentioned in this paper were written in the C language. The experiments were conducted on an IBM PC.

Table 1
Effect of changing the number of graphs used (for $n = 20$)

h	#solved	#starts	K
5	7	676	0.3
10	0	2000	4.3
15	2	1804	5.4
20	0	2000	6.1
30	7	1150	1.8
40	10	444	0
60	10	447	0
80	10	539	0
100	10	221	0
200	10	248	0

Table 2
Effect of changing upper bound on the size of graphs (for $n = 20$)

m_{upper}	h	#solved	#starts	K
3	30	9	478	0.1
5	20	6	1335	0.3
7	20	2	1694	2.0
9	20	2	1768	3.4
11	15	1	1908	3.0
13	15	3	1749	2.7
15	15	0	2000	6.8
17	15	1	1889	5.0
19	10	0	2000	4.3

The goal of the first two experiments was to evaluate the difficulty (for multi-start descent, of course) of test instances produced by GENERATE_INSTANCE by changing, in one case, the number h of graphs selected and, in another, the upper bound m_{upper} on the size of graphs. The results for $n = 20$ are summarized in Tables 1 and 2. The second column of Table 1 gives the number of instances of the QAP solved to optimality (for each row 10 instances were tried). For each instance, the multi-start descent procedure was run until either 2000 starts were performed or until an optimal solution was found. The actual number of starts of descent averaged over 10 instances is given in the column under heading "#starts". The last column displays the average value of K. The same columns are also present in Table 2. The values of other parameters to the generator are as follows: $n_x = n_y = 7, w = 10, \rho_{\text{bound}} = 50, m_{\text{lower}} = 3$, and, for the first table, $m_{\text{upper}} = 19$.

Table 3
Input data

n	t_{GEN2}	t_{rest}	h_5	h_7	h	grid dimensions	
						GEN0	other generators
20	50	80	10	5	10	5×4	7×7
30	100	200	20	10	20	6×5	8×8
40	200	350	30	15	20	8×5	9×9
50	300	600	50	20	30	10×5	10×10
60	450	900	80	30	30	10×6	11×11
70	600	1200	120	40	50	10×7	12×12

Table 4
Number of instances solved optimally (25 instances tried for each entry of the table)

Generator	Problem size; number of starts of descent					
	20; 5000	30; 2000	40; 1000	50; 500	60; 200	70; 100
GEN0	25	25	25	24	25	20
GENLP	25	25	25	25	14	8
GEN1	25	25	22	20	11	3
GEN2	25	19	20	11	8	4
GEN	2	0	0	0	0	0

The results in Table 1 indicate that the most difficult instances are obtained for rather (but not too) small number of graphs. As this number increases the instances become easier. Table 2 shows that it is reasonable to use larger values of m_{upper}, for example, $m_{\mathrm{upper}} = n$ – such a choice, in general, leads to harder QAP instances.

In the third experiment we tried to compare, using multi-start descent, the hardness of QAP test cases produced by different generators – those listed above in this section and GENERATE_INSTANCE. The input data used are given in Table 3. In this table, t_{GEN2} is the number of triangles to be selected in GEN2, t_{rest} is the same characteristic for GEN0 and GEN1, h_5 (respectively, h_7) is the number of graphs isomorphic to H_5 (respectively, H_7) – this pair of numbers is for GEN2, and h is the number of graphs for GENERATE_INSTANCE. The last columns give grid dimensions for GEN0 and, respectively, for the rest of generators. The values of some less important parameters for GEN0-GEN2 and GENLP are the same as in the experiments whose results are summarized in Palubeckis (1999). For the algorithm described here, $m_{\mathrm{lower}} = 3, m_{\mathrm{upper}} = n-1$ and $w = 10$. The results obtained are reported in Table 4. The name GEN (see the last row) is used to denote GENERATE_INSTANCE. For each value of n and corresponding number of starts of descent 25 test cases were created by each generator. The table shows how many of them were solved to optimality. It provides an ample evidence that instances

produced by GENERATE_INSTANCE are much harder than those constructed by other generators.

Table 5 contains the results of longer runs of multi-start descent on 12 test cases Inst20, Inst22,..., Inst200 created by GENERATE_INSTANCE and, for comparison, four Nugent-type problems from the QAPLIB: Nug20, Nug22, Nug24 and Nug30. The dimension of each problem is equal to the number appearing in its name. As before, the multi-start descent procedure was run until either a specified number of starts was performed or until an optimal solution was found (for Nug20, Nug22, Nug24 and twice for Inst20). The actual number of starts is given in the second column of the table. The third column reports the solution time in the form hours:minutes:seconds or minutes:seconds or just seconds. Next two columns display the optimal value and, respectively, the value for solution obtained by multi-start descent. In the case of Inst20,..., Inst200 the values are computed assuming that the flow matrix W is made symmetric by assigning w_{ij} to w_{ji} for each pair $i, j, i < j$. The last column gives the value of the optimality measure K. For each of Inst20-Inst30 and Nug20-Nug30 multi-start descent was applied three times, so these problems are represented in the table by three consecutive rows. In this experiment we run our generator with values of some parameters different than in the previous one: we took $m_{\text{lower}} = n - 1 = m_{\text{upper}}, h = n/2$, and for Inst200 $w = 5$. It seems that by taking larger m_{lower} we have increased the difficulty of instances generated.

Comparing Inst20-Inst30 and Nug20-Nug30 we can conclude that test cases produced by the algorithm presented in this paper are more difficult than the well-known Nugent-type benchmark problems from the QAPLIB. For example, Nug22 already succumb to 1000 iterations, whereas for Inst22, the QAP instance with the same number of objects, even 250000 of starts are insufficient. In fact, Nug20, Nug22 and Nug24 were solved quite easily. However, multi-start descent heuristic applied with the upper bound of 100000 on the number of starts failed to find an optimal solution for both Nug30 and Inst30. Nevertheless, the solutions obtained for Nug30 are much closer (in the sense of K) to the optimum than those for Inst30.

The distance and flow matrices defining Inst20, Inst30,..., Inst200 used in this last experiment are publically available at `http://www.soften.ktu.lt/~gintaras/qproblem.html`.

4. Conclusions

In this paper we described an algorithm for generation of test cases for the QAP. The algorithm uses special signed graphs having nonnegatively valued bisections. It differs from similar existing generators in that the sizes of such graphs are larger – up to the dimension of the problem. The set of possible outputs of this algorithm is hard, that is, no polynomial-time approximation algorithm for the QAP exists, unless P=NP, which can solve each instance in this set.

As it follows from experimentation with the multi-start descent heuristic, the test cases generated are hard not only theoretically but also in practice. Quite believable that they remain hard for other more sophisticated iterative techniques for solving the QAP. It would

Table 5
Results of longer runs of multi-start descent

problem	#starts	time	optimal value	value obtained	K
Inst20	79493	26:23	81536	81536	0.0
	148804	49:27	81536	81536	0.0
	300000	1:39:30	81536	81548	0.3
Nug20	286	6	2570	2570	0.0
	152	3	2570	2570	0.0
	219	5	2570	2570	0.0
Inst22	250000	1:43:42	107016	107184	3.6
	250000	1:43:38	107016	107084	1.5
	250000	1:43:35	107016	107268	5.4
Nug22	782	30	3596	3596	0.0
	462	18	3596	3596	0.0
	77	3	3596	3596	0.0
Inst24	200000	1:58:12	153144	153464	4.6
	200000	1:58:11	153144	153436	4.2
	200000	1:58:12	153144	153216	1.0
Nug24	2212	1:40	3488	3488	0.0
	2454	1:50	3488	3488	0.0
	1950	1:27	3488	3488	0.0
Inst30	100000	2:08:37	271092	272080	8.0
	100000	2:08:36	271092	272128	8.4
	100000	2:08:44	271092	272300	9.8
Nug30	100000	2:58:15	6124*	6128	0.2
	100000	2:58:15	6124*	6130	0.3
	100000	3:00:40	6124*	6128	0.2
Inst40	50000	2:58:39	837900	840308	10.0
Inst50	20000	2:41:28	1840356	1846876	13.2
Inst60	10000	2:35:51	2967464	2978216	14.2
Inst70	5000	2:12:19	5815290	5831954	14.9
Inst80	3000	2:08:30	6597966	6618290	14.8
Inst100	1500	2:16:53	15008994	15047406	14.4
Inst150	500	3:13:22	58352664	58468204	15.6
Inst200	200	3:45:13	75405684	75543960	16.0

* best known value

be interesting to test algorithms based on these techniques on the instances generated by our algorithm, particularly on those of type used in the last experiment of the previous section.

References

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

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

Burkard, R.E., E. Çela, P.M. Pardalos and L.S. Pitsoulis (1998). The quadratic assignment problem. *SFB Report 126*, Institute of Mathematics, Technical University Graz, Austria, to appear in P.M. Pardalos and D.-Z. Du (Eds.), *Handbook of combinatorial optimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

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

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

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

Clausen, J., S.E. Karisch, M. Perregaard and F. Rendl (1996). On the applicability of lower bounds for solving rectilinear quadratic assignment problems in parallel. *Technical Report DIKU-TR-96/24*, Department of Computer Science, University of Copenhagen, Denmark, to appear in *Computational Optimization and Applications*.

Clausen, J., and M. Perregaard (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 Journal of Operational Research*, **46**, 93–100.

Cung, V-D., T. Mautor, P. Michelon and A. Tavares (1996). A scatter search based approach for the quadratic assignment problem. *Technical Report No.96/037*, Université de Versailles, France.

Cyganski, D., R.F. Vaz and V.G. Virball (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.

Elshafei, A.E. (1977). Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, **28**, 167–179.

Fleurent, C., and J.A. Ferland (1994). Genetic hybrids for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **16**, 173–187.

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

Geoffrion, A., and G. Graves (1976). Scheduling parallel production lines with changeover costs: practical application of a quadratic assignment/LP approach. *Operations Research*, **24**, 595–610.

Hubert, L. (1987). *Assignment Methods in Combinatorial Data Analysis.* Marcel Dekker, Inc., New York.

Krarup, J., and P. Pruzan (1978). Computer-aided layout design. *Mathematical Programming Study*, **9**, 75–94.

Laporte, G., and H. Mercure (1988). Balancing hydraulic turbine runners: a quadratic assignment problem. *European Journal of Operational Research*, **35**, 378–381.

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

Li, Y., P.M. Pardalos and M.G.C. Resende (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 C. Roucairol (1994). A new exact algorithm for the solution of quadratic assignment problems. *Discrete Applied Mathematics*, **55**, 281–293.

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

Nugent, C.E., T.E. Vollmann and J. Ruml (1968). An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, **16**, 150–173.

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.

Palubeckis, G. (1999). Generating hard test instances with known optimal solution for the rectilinear quadratic assignment problem. *Journal of Global Optimization*, **15**, 127–156.

Pardalos, P.M., F. Rendl and H. Wolkowicz (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., M.G.C. Resende and Y. Li (1996). FORTRAN subroutines for approximate solution of dense quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, **22**, 104–118.

Pardalos, P.M., M.G.C. Resende and L.S. Pitsoulis (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.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal 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 A.E. Smith (1995). A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, **22**, 73–83.

Wilhelm, M.R., and T.L. Ward (1987). Solving quadratic assignment problems by simulated annealing. *IEEE Transactions*, **19**, 107–119.

G. Palubeckis received the degree of Doctor of Sciences from the Kaunas Polytechnic Institute, Kaunas, Lithuania, in 1987. Currently, he is an Associate Professor at the Kaunas University of Technology. His major research interests are in graph theory, combinatorial optimization, and computational geometry.

Testinių pavyzdžių kvadratinio paskirstymo uždaviniui konstravimo algoritmas

Gintaras PALUBECKIS

Straipsnyje pateikiamas algoritmas kvadratinio paskirstymo (KP) uždavinio pavyzdžių su žinomu optimaliu sprendiniu generavimui. Tokių pavyzdžių srautų matrica yra konstruojama imant matricas atitinkančias specialiems grafams, kurių dydis gali pasiekti uždavinio apimtį. Šiuo atžvilgiu algoritmas apibendrina kai kuriuos egzistuojančius algoritmus, paremtus tiktai daugkartiniu trikampių išrinkimu. Aibė pavyzdžių, kurie gali būti suformuoti algoritmo pagalba, yra NP-sunki. Naudojant daugelio startų nusileidimo euristiką KP uždaviniui, šie testiniai pavyzdžiai yra eksperimentiškai palyginami su pavyzdžiais, formuojamais kelių egzistuojančių generatorių, o taip pat su Nugento-tipo uždaviniais, paimtais iš KP uždavinio bibliotekos QAPLIB.